FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Discovery of Requirements Dependencies through Web Usage Analysis

Mailson Gabriel Rodrigues da Silva

MEINF - MASTER IN INFORMATION ENGINEERING

Supervisor: Ana Cristina Ramada Paiva (PhD)

Co-Supervisor: Jorge Manuel Esparteiro Garcia (PhD)

July 2017

Discovery of Requirements Dependencies through Web Usage Analysis

Mailson Gabriel Rodrigues da Silva

MEINF – Master in Information Engineering

Aprovado em provas públicas pelo Júri:

Presidente: Antonio Pedro Rodrigues Aguiar (PhD)

Vogal Externo: Miguel António Sousa Abrunhosa Brito (PhD)

Orientador: Ana Cristina Ramada Paiva (PhD)

---

19 de Julho de 2017

Abstract

With the improvement of the technologies and frameworks available to create Cloud-based apps, it is now possible to develop, deliver and maintain applications that have high availability and reach a wide range of users. To maintain such web applications can be a challenge, for instance, an update or new feature has to keep the website available for the end user. To keep track of user's behavioural patterns, websites often collect information about their usage, analyse this data and produce recommendations to improve websites and help their maintenance. REQAnalytics is a recommender system that collects the information about the usage of a website, processes it, and generates recommendations to the requirements specification of the website, such as requirement prioritization, functionalities to create or remove, and traceability between website elements and functionalities [Garcia & Paiva, 2016]. This paper proposes an evolution to the REQAnalytics system, in which we collect the dependencies between requirements through a Requirements Interchange Format (ReqIF) [Group, 2016] and then merge them with the information about the website usage. With this correlation, based on the user session information, it is possible to detect not used or not defined dependencies, and produce new recommendations to the requirements specification. Another possibility is to support the impact analysis of proposed changes, helping to maintain the software requirements specification updated and traceable. A case study validates the proposed extension.

Acknowledgements

Firstly, I would like to thank God for the inspiration and discernment that took me to this point of my life.

Secondly, my mother Iracy, who is a real warrior of the real life, who raised me all by herself and gave me the conditions to pursuit my dreams, and still supporting me with life advices, thank you.

To Ana Maria, my partner is this journey, my sincere gratitude and all my love.

A special mention to the Erasmus Mundus programm for providing the scholarship and support during the two years.

Finnaly, to my supervisors, Ana Paiva and José Garcia, thank you for leading me through your experience and knowledge, and for letting me dig into your awesome tool.

Mailson Gabriel Rodrigues da Silva

Contents

List of Figures

List of Tables

List of Abbreviations

HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
PHP	Hypertext Pre-processor. Open Source script language
RE	Requirements Engineering
RM	Requirements Management
RT	Requirements Traceability
RTM	Requirements Traceability Matrix
SAAS	Software as a Service
SOA	Service Oriented Architecture
SDLC	Software Development Life Cycle
SRS	Software Requirements Specification
RIF/ReqIF	Requirements Interchange Format
ULS	Ultra-large-scale
UI	User Interface
XML	extensible Mark-up Language
WWW	World Wide Web

Chapter 1

Introduction

Cloud-based services are continuously updated and improved, in this way, keeping the requirements specification documents updated and aligned to the implemented services can be a challenge. However, information about the current state of functionalities and its dependencies is crucial and useful to assess the impact of changes and play an important role in the overall service quality. Nowadays websites are seen as an essential tool for business development, they are used as Web Services, Intranets and Web applications [Singh & Singh, 2010], [Taghipour, Kardan, & Ghidary, 2007]. Most of the web applications have to be available 24 hours per day and, to satisfy its customers, the quality of this type of software is a critical concern. Besides that, web applications should be maintained and improved during the software's lifecycle, so that the stakeholder's needs may be fulfilled [Malviya & Agrawal, 2015].

Many benefits come with the use of Software as a Service (SaaS) and Web Services integrated with a Service Oriented Architecture (SOA), namely interoperability, security, and the ability to adapt quickly to the usual changing business demands. However, its use adds a new complexity layer to the environments where these services are deployed [Barbir et al., 2007]. In terms of complexity some challenges arise, such as the replace ability, compatibility, and process conformance of exchanged services, where the data have to be preserved and formally validated without losing its meaning [Papazoglou et al., 2007].

To achieve a significant users' satisfaction, companies must continuously dedicate efforts to maintain and improve the quality of the released web applications. It can be realized applying tools and methodologies that can guarantee the quality of the development process and, consequently, the quality of the product [Hariguna, Lai, & Chen, 2016]. In this context emerges

REQAnalytics, a recommender system that collects information about the usage of a web application, relates it back to the requirements, and generates reports with recommendations and change suggestions that can increase the quality of the web application. The system provides reports in a business-friendly language where, for example, it shows new workflows and navigation paths, identifies features to remove and presents the relationship between requirements and the proposed changes helping to maintain the software requirements specification updated and useful.

Recommender Systems for Software Engineering (RSSE) [Walker & Zimmermann, 2014] arise from the necessity of automation of procedures among software engineering users, where the recommendations help software developers, for example, these tools can suggest code reuse [Ye & Fischer, 2005], support software requirements elicitation [Castro-Herrera & Cleland-Huang, 2010] and communicate user input to the engineering team.

Web usage mining can gather input data for the recommender system, working with real and live information about user behaviour on a web application. Combining a recommender system, web usage mining and requirements engineering can produce information that will assist the requirements maintenance. In particular, it is common to have information about requirements' dependencies outdated, which can influence the analysis for changes. This paper presents an approach capable of detecting requirements' dependencies from usage data automatically gathered.

1.1 Context

In Software Engineering, one of the biggest concerns has to do with software quality, i.e., ensure that software is systematically, rigorously, measurably, on time, on budget, and within specification built.

As software evolve along time, there is the need to define new requirements, update existing ones and implement changes accordingly. Change requests may originate from different stakeholders, such as developers, systems engineers, users and service integrators. Changes in requirements, their uncontrolled evolution throughout the project and a weak requirements elicitation are the main causes for software project failures [Standish Group, 2015].

Managing requirements helps to prevent the negative impacts of the uncontrolled requirements changes and assure that the stakeholder's needs are fulfilled. By means of the analysis of change effect and by auditing it, may be accomplished a proper requirements

management [Ibrahim, Kadir, & Deris, 2009]. However, keep a Software Requirements Specification updated is difficult, particularly on cases where the software project is a website/service that evolves during its lifetime. This happens, despite of software organizations are trying to improve their processes used to collect, analyse, document and maintain their requirements in a structured requirements specification written in natural language [K. E. Wiegers, 1999]. As times goes by a deficient requirements management process leads to obsolete requirements that do not express the actual state of development of websites becoming useless.

Another difficulty through software lifetime is defining the correct priority for change requests' implementations and assess the impact of those changes because often there is no traceability information between the requirements and implementation neither are identified the dependencies among different requirements.

During the Software Development Life Cycle (SDLC) it is mandatory that a good Requirements Management process is put in place, special attention should be dedicated to the requirements documents and its quality [Rupp & Pohl, 2011]. A good quality requirement document must be unambiguous and consistent, have a clear structure, be modifiable and extendable, be complete, and traceable [Rupp & Pohl, 2011].

The quality criteria 'traceability' is the most important for the context of this study, where the focus is to keep an updated track of dependencies among requirements and to detect new dependencies not previously defined.

1.2 Problem and Motivation

There is a split in modern software development. In one hand, professionals with knowledge of programming, project management and database carry out the conventional software development. In other hand, however, the web development aggregates a wide range of developers, who don't necessarily need programming skills to create and deploy websites, it can be done without having to write a single line of code [Mendes, 2014]. In this web environment, also called SaaS (Software as a Service), where updates and maintenances are frequent, having a traceability matrix with the dependencies among the updated requirements is a key factor to maintain the web application running without impacting the user experience [A. Garcia & Paiva, 2014].

This paper aims to show that extracting and analysing data about the usage of websites can help to maintain requirements updated, update dependencies among requirements and contribute to improve the overall quality of the services.

Web Analytics is a set of methods for measurement, data collection, and data analysis of websites during their lifetime. Its main purpose is to understand and optimize the web usage. The usage of websites generates large amounts of information that have different purposes, like assessment of quality of web-products [Pai, Ravindran, Rajagopalan, & Srinivasaraghavan, 2013] or to statistical analysis of website data usage [Kumar, Singh, & Kaur, 2012]. Data from Web Analytic tools together with a proper analysis of that data may produce recommendations to improve the overall software service quality.

Web Analytics tools are able to extract several data about the usage of a website. The gathered data produce diverse information that helps to evaluate, for instance, the quality of web products [Singal, Kohli, & Sharma, 2014], pattern recognition or to statistical analysis of website data usage[Kumar et al., 2012].

Current requirements management methods do not use this type of information, having its main focus on the analysis and reporting of business metrics like number of visits and traffic sources, these tools are more interesting to marketers [Pai et al., 2013] and therefore still has some limitations for the maintenance of services. The possibilities of the analysis of this web usage data still need to be explored [Akerkar, Bădică, & Burdescu, 2012].

Some challenges on the Web Analytics field need to be addressed, for example, there are several web analytics tools that collect large volumes of data, but they do not give any recommendations to the improvement of the website based on the data collected. In addition, the stakeholders are often sceptical of a new form of automated tool support [Thurimella & Maalej, 2013]. Therefore, high quality recommendations are necessary to answer the stakeholder's doubts. There is also no focus on the improvement of the Software Requirements Specification (SRS).

The web analytics tools currently available have a number of weaknesses. For instance, they do not analyse the service based on different user roles; do not analyse typical navigational paths; do not produce reports based on the requirements.

1.3 Objectives

The main objectives for this report are:

- Successfully import requirements in the ReqIF format;
- Identify and recommend the creation of new dependencies from the web usage data;

1.4 Report Organization

This paper is organized in five chapters. Chapter 2 presents a discussion about the State of the Art on Requirements Engineering, Requirements Management, Software as a Service (SaaS), Web Usage mining, Recommender Systems, and the related work; it also describes the REQAnalytics recommender system. Chapter 3 proposes an approach for the evolution of the existing REQAnalytics system, firstly is demonstrated an extension to automatically import functional requirements XML documents; secondly an extension that uses the functional requirements dependencies and its related user session's data, collected by a Web Analytics tool, to detect new, undefined, functional requirements dependencies. In Section 4, we present a detailed Case Study, where the approach proposed in Section 3 is applied and the results are discussed. Chapter 5 presents the conclusions and future work.

Chapter 2

State of the Art

Almost every company today uses at least one software, from the small and simple sales management system of your local grocery store to the very large systems of a multinational corporation, or even in your personal life with your mobile phone, for example. We are connected and relying on software to help us in the daily activities.

With this in mind is of extreme importance that we, the software community, keep doing our jobs in the best possible way, providing solid, strong, and stable software products. To achieve this commitment with quality a proper application of Software Engineering skills is mandatory.

Software engineers, analysts, developers, experts, etc., are responsible for translating user and stakeholder's needs into a product that will deliver a solution for their necessity.

Requirements Engineering provides the methods for elicitation, specification, validation, and management of user's needs, which will be detailed in this section, along with related works on Impact Analysis, Web Usage Mining and Recommender Systems.

2.1 Requirements Engineering

Requirements Engineering (RE) is defined as the process of eliciting, documenting, and maintaining individual stakeholder's requirements and needs, developing them into detailed requirements which are documented and specified to serve as the basis for all other software development activities [Pohl, 2010].

In software engineering, such requirements documents are often called Functional Requirements Specifications. The Functional Requirements Specifications is the formal response

to the defined objectives. It describes all external user and programming interfaces that the product must support.

Sommerville [Sommerville, 2010] defines functional requirement as:

> *These are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organisation when writing requirements. When expressed as user requirements, the requirements are usually described in a fairly abstract way.*

The classic definition of what a requirement is can be taken from the IEEE Standard 610.12 of 1990 [Ieee, 1990]:

- A condition or capability needed by a user to solve a problem or achieve an objective;
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents;
- A documented representation of a condition or capability as in the previous items.

Alternatively, a more modern definition is given by [Glinz, 2014] as:

- A need perceived by a stakeholder;
- A capability or property that a system shall have;
- A documented representation of a need, capability or property.

The challenges faced in requirements engineering are detailed by many authors in related works. An example is the lack of communication and misunderstanding of requirements specifications [Bubenko, 1995], especially in large software projects, where breakdowns in communication can occur more easily [Al-Rawas & Easterbrook, 1996]. Poor communication still one of the most common problems in eliciting customer's requirements [Pa & Zin, 2011].

The physical location of the stakeholders and engineering teams [Salvatore & Alameda, 2003], and multi-site software development organisations [Damian, 2007], are challenges related to location, but also impact the communication when the channels are not properly defined.

Some authors tackle the agile software development process. They write about how to apply traditional requirements engineering methods in agile requirements engineering [Inayat, Salim, Marczak, Daneva, & Shamshirband, 2015], and also how to perform requirements traceability in agile software development [De Lucia & Qusef, 2010].

With the immense number of requirements engineering tools available in the market, it is of extreme importance to have some sort of integration amongst them, in this scenario XML-based solutions should be considered to create such tool integrations [Meisenbacher, 2005]. An example of XML-based integration is given is this dissertation (see Section 3.2.1).

Ultra-large-scale (ULS) systems also create a challenge for the RE community, due to its scale and complexity [Ncube, 2011]. ULS systems "comprises not only information technology (IT) components, but also machines of many kinds, individuals and teams, diverse sensors, information streams and stores (including verbal and non-verbal human communications), and so forth" [Northrop et al., 2006]

2.1.1 Requirements Engineering Activities

Requirements Engineering is divided in four main activities [Pohl, 2010]:

- Elicitation: the process of obtaining requirements from stakeholders, using different techniques;
- Documentation: this activity uses natural language or conceptual models to describe the requirements elicited previously;
- Validation and Negotiation: documented requirements must be validated and negotiated early on, to guarantee that all the predefined quality criteria are met;
- Management: responsible for dealing with the requirements documents and artefacts, it runs together with all other activities, is responsible for maintaining the consistency after changes, and ensure their implementation.

Figure 1 shows a spiral model view of the requirements engineering activities.

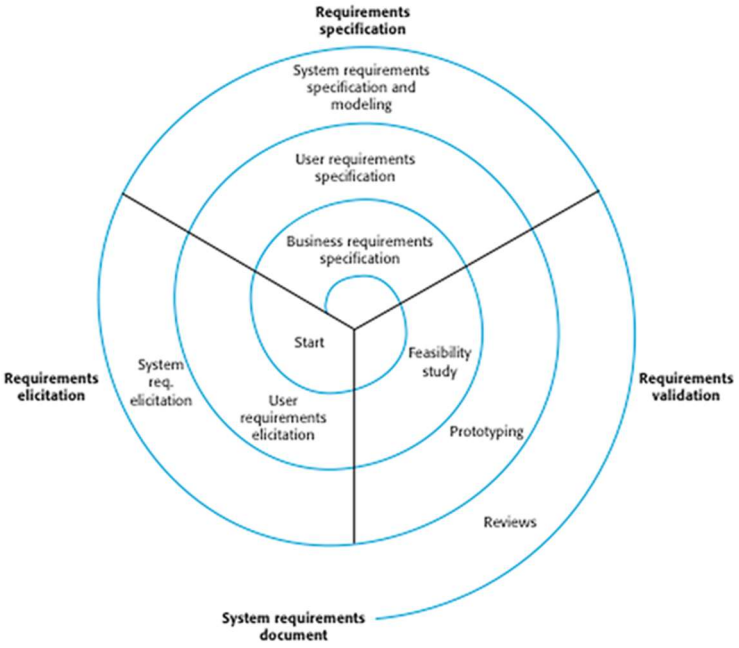


Figure 1: Requirements engineering processes

Requirements Management, which is described as a support to all other processes [Sommerville & Kotonya, 1998] , is detailed in the next section. The other activities are not part of the scope of this dissertation.

2.1.2 Requirements Management

Requirements management (RM) is the process of documenting, analysing, tracing, prioritizing, and agreeing on requirements and then controlling change and communicating to relevant stakeholders [Rupp & Pohl, 2011]. It is a continuous process developed throughout the project. A requirement is a capability to which a project outcome (product or service) should conform.

Thus, software organizations that wish to have their systems well maintained should necessarily implement requirements management activities. Systems will change and evolve, keeping track of the evolution process will influence the quality of the delivered product.

Requirements management is one of the two cross-sectional requirements engineering activities, together with Requirements Validation.

The goals and activities of management in requirements engineering are [Pohl, 2010]:

1. Observe the system context to detect context changes.

The goal of this activity is to identify changes in the context and estimate the impact of these changes. Detecting such context changes is a prerequisite for analysing these changes and for incorporating them into the requirements engineering process [Pohl, 2010]. Examples of context changes:

- A new technology or a new competing product emerges;
- A law or standard changes;
- Evolution of stakeholder goals;
- Involvement of additional stakeholders;
- Change of an organisation policy;
- Changes in the way that external actors (stakeholders or systems) use the system.

2. Manage the execution of requirements engineering activities.

Aims to monitor, control, and adjust the planned workflow of elicitation, documentation, negotiation, and validation activities, applying techniques from project management. It can have a phase-oriented approach, based on waterfall process model, as shown in Figure 2, or a situative approach, where the activity to be executed next depends on the status of the requirements artefacts and the current process situation [Pohl, 2010].

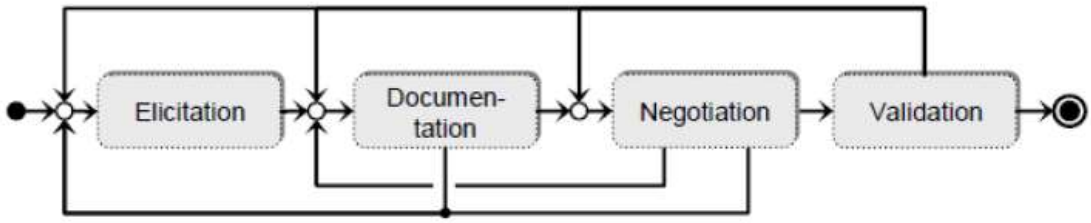


Figure 2: Phase-oriented approach for Requirements Management

3. Manage the requirements artefacts (traceability, version and configuration).

The aim of this activity is to keep track continuously of all requirements artefacts, their relevant attributes and relationships as well their evolution. Has five main sub-tasks:

- Definition of requirements attribute scheme;
- Requirements traceability;
- Requirements change management;
- Requirements configuration management;
- Requirements prioritisation.

Managing requirements helps to prevent the negative impacts of the uncontrolled requirements changes and assure that the stakeholder's needs are fulfilled. By means of the analysis of change effect and by auditing it, a proper requirements management may be accomplished [Ibrahim et al., 2009]. However, keep a Software Requirements Specification updated can be difficult, particularly on cases where the software is a website/service that evolves during its lifetime. This happens, despite of software organizations are trying to improve their processes used to collect, analyse, document and maintain their requirements in a structured requirements specification written in natural language [K. E. Wiegers, 1999]. As time passes, a deficient requirements management process leads to obsolete requirements that do not express the actual state of the websites, becoming useless.

One of the challenges in RM is to have requirements documentation that are easy to read, navigate, query, and change. Requirements Traceability (RT) is used to achieve such conditions [Nuseibeh & Easterbrook, 2000]. RT tools and activities are the basis for most tools that perform impact analysis of changes. The changes in requirements, their uncontrolled evolution throughout the project and a weak requirements elicitation are the main causes for software project failures [Standish Group, 2015].

2.1.3 Requirements Management in SaaS

Software as a Service (SaaS) is defined as "a software that is owned, delivered and managed remotely by one or more providers. The provider delivers software based on one set of common code and data definitions that is consumed in a one-to-many model by all contracted customers at any time on a pay-for-use basis or as a subscription based on use metrics." [Gartner, 2017].

SaaS is a way of providing software on demand, over the internet, by charging its real use, and not per license [Candan, Li, Phan, & Zhou, 2009]. The traditional software is sold per license, where a company or user pays to have the rights to use the software, regardless of the amount of use. In SaaS, the buyers of the software become subscribers of a service accessed in the Internet

[Liao & Tao, 2008]. Developing the software solutions using SaaS enables the software vendors to be ahead of the market innovation, bringing benefits to its customers and suppliers [Laplante, Zhang, & Voas, 2008]. In SaaS, a software is deployed as a service hosted in a remote server and accessed over the Internet [Nandi & Banerjee, 2013]. The change from traditional locally installed software to the SaaS model brings challenges for the requirements engineers, it is a paradigm shift that requires revaluation of current methodologies [Babar, Ramzan, & Ghayyur, 2011]. There is still room to grow the maturity level of SaaS and to detail its activities [Kang et al., 2010].

The concept of SaaS relies on two other recent developed technologies, first one is the Cloud Computing, which is the idea of utilizing a wide range of applications through the internet, anywhere and independent of the chosen platform, with the same ease of use as if they were locally installed [Velte, Velte, & Elsenpeter, 2010]. Second is the open concept of Service Oriented Architecture (SOA), an architectural concept that allows the development of web applications that are interoperable, can be easily reused and shared across applications [Li & Wu, 2009]. In this context, Software as a Service (SaaS) based in SOA, has become extremely relevant [Rhoton, 2010].

Using SaaS, a vendor can deliver a software system as a service. Using SOA enables the service to be published, discovered, and adopted as a component to construct new software systems, which can also be published and delivered as new services. In other words, the two models complement each other: SaaS helps to offer components for SOA to use, and SOA helps to quickly realize SaaS [Laplante et al., 2008].

Some examples of SaaS software are the Google Apps (Gmail, Calendar, Drive, etc.), Salesforce.com (ERP/CRM), Azure Web Services and Office 365 from Microsoft, and Amazon Web Services, which includes a full IT services infrastructure on the cloud with more than 70 different web services.

The use of SaaS enables the collaboration among software and services vendors, because SaaS has all the advantages of Cloud Computing and SOA architecture [Cancian, 2013]. The lack of information about the quality of the service provided by the vendors is a challenge that affects negotiation [Beil, 2009]. The information about service quality is of extreme importance for its acceptance, which is why traditional software methodologies have been applied as a way to improve the overall quality of the software system developed, with satisfactory results [Yang, Wang, & Li, 2009]. In software development, the quality of the product is directly related to the quality of the development process [Humphrey, 1988]. Therefore, it is common that the pursuit for a high quality software passes through an improvement of the development process [Beecham et al., 2005].

Thus, the SaaS software development processes could be improved, increasing the maturity and the capacity of its processes, and thereby providing more information about the software service product, increasing the acquisition confidence and the negotiation power of the vendors [Cancian, 2013].

As there is no standard maturity model or framework specially designed for the SaaS architecture, traditional requirements management activities may be applied and, when necessary, adapted to guarantee the quality of software deployed as SaaS. It is important to apply requirements management methodologies and techniques to allow the development of SaaS with the necessary agility and to ensure their prompt adaptation to change [A. Garcia & Paiva, 2014].

One of these techniques is Requirements Traceability (RT), which is detailed further in the text. To have requirements documentation with detailed RT is a way of achieving integrity and completeness of that documentation, playing an important role in impact analysis and change management [Nuseibeh & Easterbrook, 2000].

2.1.4 Requirements Traceability

Requirements Traceability can provide many benefits to organizations that properly use traceability techniques [Kannenberg & Saiedian, 2009]. Requirements Traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction [Gotel & Finkelstein, 1994]. The goal of forward traceability is to make sure that each requirement is implemented in the product and that each requirement is completely tested [Westfall, 2006]. Backward traceability can verify if the requirements have been kept current with the design, code, and tests [Westfall, 2006]. Requirements Traceability Matrix (RTM) is commonly used as a method for tracing requirements to their outcomes. A RTM is defined as a table that illustrates logical links between individual functional requirements and other system artefacts [K. Wiegers & Beatty, 2003]. However, manual traceability methods may not be feasible because traceability links that need to be captured grows exponentially with the size and complexity of the software system [Cleland-Huang, Chang, & Christensen, 2003].

Requirements Traceability is recognized as a critical success factor in software development [Dömges & Pohl, 1998] and has been recognized as an important quality of a well-engineered software system [Gotel & Finkelstein, 1994]. Requirements Traceability is an important method during the maintenance phase because the initially defined requirements often change during the life cycle of the software development, and it is very important to assess the impact of those changes. Traceability allows determining what requirement, test cases or other artefacts, need to

be changed and can determine the costs and risks associated with that change. [Sherba & Anderson, 2003] added another question that a traceability approach should also answer: how traceability relations will be viewed and queried. Traceability analysis is related to the process of tracking forward or backward through a network of interrelationships between components of a System and its documentation [“Traceability Analysis,” n.d.].

A survey found that 20% of the requirements were involved in roughly 75% of all interdependencies [Carlshamre, Sandahl, Lindvall, Regnell, & Dag, 2001]. There are different approaches and mechanisms in the literature that aim to achieve traceability maintenance [Mäder & Gotel, 2012]. They can be classified into: Subscription-based approaches [Cleland-Huang et al., 2003], Rule-based approaches [Spanoudakis, Zisman, Pérez-Miñana, & Krause, 2004] [Murta, Van Hoek, & Werner, 2006] and approaches based on recognizing evolution [Cleland-Huang et al., 2003].

A proper Requirements Traceability leads to a more qualified and cost-effective Impact Analysis of the change requests.

Impact Analysis

Having traceability information updated is useful to perform impact analysis when there are change requests and help to implement those changes without degrading the quality of the website. Impact Analysis is defined as "identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change" [Bohner & Arnold, 1996].

Bi-directional links between functional requirements allow to determine which derived requirements, or parts, might be affected by a change in customer requests [Ramesh, Powers, Stubbs, & Edwards, 1995].

Impact analysis can be separated in three methods [Kilpinen, 2008]:

- Traceability: Links between requirements, specifications, design elements, and tests are captured, and these relationships can be analysed to determine the scope of a proposed change [Bohner & Arnold, 1996].

- Dependency: Links between parts, variables, logic, modules, etc. are assessed to determine the consequences of a proposed change. Dependency occurs at a more detailed level than traceability [Bohner & Arnold, 1996].

- Experiential: The impact of changes is determined using experts knowledge [Hassine, Rilling, Hewitt, & Dssouli, 2005].

None of the methods cited above utilizes automation to collect the traceability amongst requirements and other artefacts necessary, either an expert knowledge or previous documentation is necessary to perform the task of impact analysis.

There is where the discipline/field of study on Web Usage Mining can be useful, to automate the process of collecting traceability based on the real usage of a website. Further details of this area are discussed next.

2.2 Web Usage Mining

Web Usage Mining plays an important role in usability of the website, the improvement of user's relations and improving the necessity of system presentation [Malviya & Agrawal, 2015]. One important kind of data is the data generated by users' click stream. This data can be useful to predict user behaviour and to redirect the user to the required information more easily.

Web usage mining can be described as the process of collection and discovery of user access patterns from web usage logs, it has a great potential for recommendations based on users preferences [Singal et al., 2014]. [Srivastava, Cooley, Deshpande, & Tan, 2000] described web usage mining as "the application of data mining techniques to discover usage patterns from Web data, in order to understand and better serve the needs of Web-based applications".

The process of web usage mining is normally divided into two main tasks:

- Data preparation. The task of recording user sessions into a file or database, where detailed information is stored for latter pattern mining [Srivastava et al., 2000];
- Pattern discovery. This task involves the discovery of association rules, sequential patterns, usage clusters, page clusters, user classifications or any other pattern discovery method [Cooley, Mobasher, & Srivastava, 1999].

Pattern discovery sums up different fields of study, from statistics to data mining, machine learning, and pattern recognition. It comprises several methods and activities, as described below [Srivastava, Cooley, Deshpande, & Tan, 2016]:

- Statistical analysis: from the usage data collected from a website, it is possible to perform statistical analysis, calculate frequency, mean, median, etc. The most frequent accessed page(s) is an example of this kind of statistical technique.

- Association rules: used to relate web pages most often accessed or referenced together in a website. Useful to reveal correlation among users and user sessions.
- Clustering: Technique to group information that have similar characteristics, for example demographic information about users. If the user sessions can be separated by age, for example, different products or services could be marketed specially to the targeted age range.
- Sequential patterns: Used to find inter-sessions patterns in order to predict future visit patterns.
- Dependency modelling: This technique aims to develop a model capable of representing significant dependencies among web variables from a website. Used to predict user behaviour, improve website navigation and increase sales of offered products.

Usage pattern extracted from web data can be applied to a wide range of applications such as web personalization, system improvement, site modification, business intelligence discovery, and usage characterization [Cho, Kim, & Kim, 2002]

Currently web usage mining tools collect information from three main sources: Web servers, proxy servers, and Web clients [Facca & Lanzi, 2004]. Having its main focus on the analysis and reporting of business metrics like number of visits and traffic sources, these tools are more interesting to marketing analysis [Pai et al., 2013].

Websites that sell products or services (e-commerce) have richer and more detailed data when compared to off-line commerce data. One type of data is the clickstream, which is the visitor's click path through the web site. Clickstream provides essential information to understanding patterns and behaviours of customers, such as what products they see, add to the shopping cart, and what products they finally buy. Through web usage mining, it is possible to make a more accurate analysis of customer's interests or preferences across all products than analysing the purchase records only [Cho et al., 2002].

The relationship between web usage mining and requirements management may provide recommendations on how to improve the web services and help maintaining requirement documents actual and useful [Garcia & Paiva, 2016].

2.3 Recommender Systems

Recommender systems are generally used in e-commerce websites to provide user personalization like product, content or service's recommendations. This class of system is called content-based recommendation system [Pazzani & Billsus, 2007]. Concepts of Information Retrieval (IR) are behind this kind of recommender systems, such as Term Frequency times

Inverse Document Frequency (TF.IDF). TF is the frequency of a word in a document and IDF is the inverse of the document frequency among the whole corpus of documents, they are used to determine the relative importance of a document and/or product/content in a website [Shuvayan, 2015]. Search engines also use TF.IDF as a way to score and rank the importance of a document for a given query. Recommendation systems are able to give recommendations because of these calculations of relevance of a determined query (which can be a user browsing for a product in an e-commerce website).

A study presented a merge of web usage data mining and recommendation systems, based on current user behaviour [Adeniyi, Wei, & Yongquan, 2016], however the approach focus on recommendations for the website regular user and not for the technical team.

Besides its most common use, recommender systems may also be used to support developers and/or software engineers in the decision making process, this class of system is called Recommender Systems for Software Engineering (RSSE).

RSSE are systems created to automate the procedures realizd by software engineering users, where the recommendations may help software developers in general [Walker & Zimmermann, 2014]. For example, these tools can suggest code reuse [Ye & Fischer, 2005], support software requirements elicitation [Castro-Herrera & Cleland-Huang, 2010] and communicate user input to the engineering team. Such systems aim to improve the system quality and the development process quality, by helping the software developers to make better decisions. They generally have code reuse and debugging features, acting in the implementation and maintenance phases [Gasparic & Janes, 2016].

As we have seen in the available literacture and research work, there is no RSSE tool currently using context information or making behavioral analysis. To supply this gap, a novel tool arises, REQAnalytics [Garcia & Paiva, 2016], which aims to improve the requirements maintenance based on the usage of a website. The usage information is gathered by a web analytics tool, mapped to the existing functional requirements of the website, thus enabling REQAnalytics to generate recommendations reports that may help the requirements maintenance and increase the overall quality of the software. Web usage mining is used to gather input data for the REQAnalytics recommendation system, working with real and live information about user behavior on a web application.

2.4 REQAnalytics

REQAnalytics is a recommender system for software engineering [J. E. Garcia & Paiva, 2016a], which aims to improve the requirements specification maintenance. The recommendations given by the system are based on web usage data gathered through a web analytics tool.

REQAnalytics maps the features available on a website to the functional requirements of the software specification. Having this mapping and gathering the usage of the system, the tool is able to give a set of recommendations. It also shows a traceability matrix between web elements and the requirements specifications. The system analyses the web usage data of a specific website and generates recommendations that may help the requirements engineers to manage and maintain software requirements specifications.

The result of this analysis is a detailed report with several recommendations to the selected website requirements specifications.

REQAnalytics analyses the web usage data of a specific website and generates recommendations to the requirements engineers. To generate these recommendations REQAnalytics has four phases, described below, and the workflow of the tool is described in Figure 3.

- First phase has two steps: step one consists in the import of the functional requirements from an XML document, this document has attributes such as ID, Description, Status, Owner, Date, and Priority Level (Low, Medium or High), indicating the priority order of a requirement's change request. Second step is the requirements mapping, where a map between the functional requirements and the implementation artefacts of the website is created, with the aid of a mapping tool (see Section 2.8) designed as a bookmarklet on a web browser.
- Second phase: An open source web analytics tool (OWA – see Section 2.9) is used to collect the web usage data from the website, saving it to a MySQL database.
- Third phase: Analysis of the web usage data collected by OWA, joining it with the mapped information generated by the mapping tool.

- Fourth phase: The result of the previous analysis is a detailed recommendation report with several recommendations to the selected website requirements specifications, with possible improvements to the functional requirements.

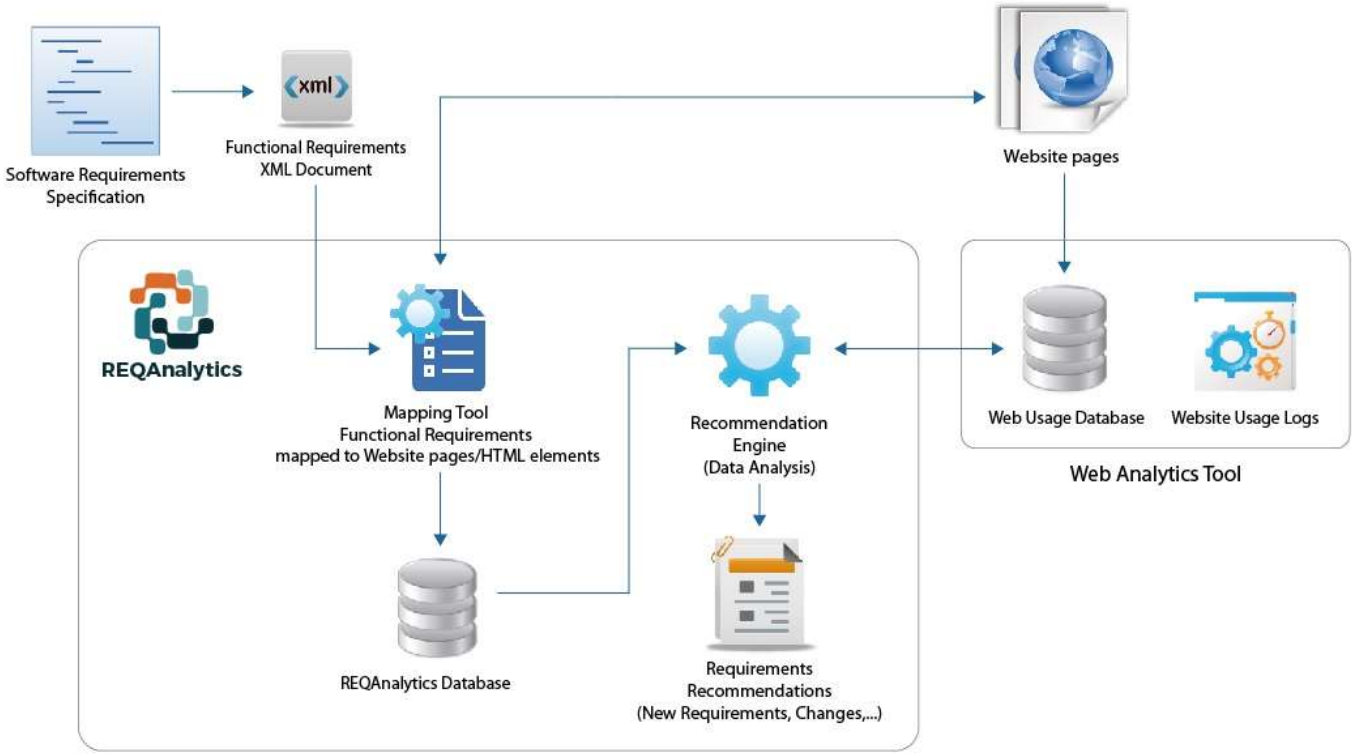


Figure 3: REQAnalytics workflow

The goal of the REQAnalytics Recommender System is to complement and support the task of Requirements Maintenance. The main features of the system are described below:

- **Mapping Tool -** Relates the functional requirements specifications of the website to the web pages and elements that implement them, thus being able to produce a traceability matrix.
- **Integration with a web analytics tool -** Reads the web usage data provided by a web analytics tool, correlates this data with the functional requirements and then gives recommendations, as seen in Figure 4, which can be to:
    - Create New Requirement.
    - Change Requirements Priority.
    - Delete Requirements.
    - Split a requirement in two or more requirements.

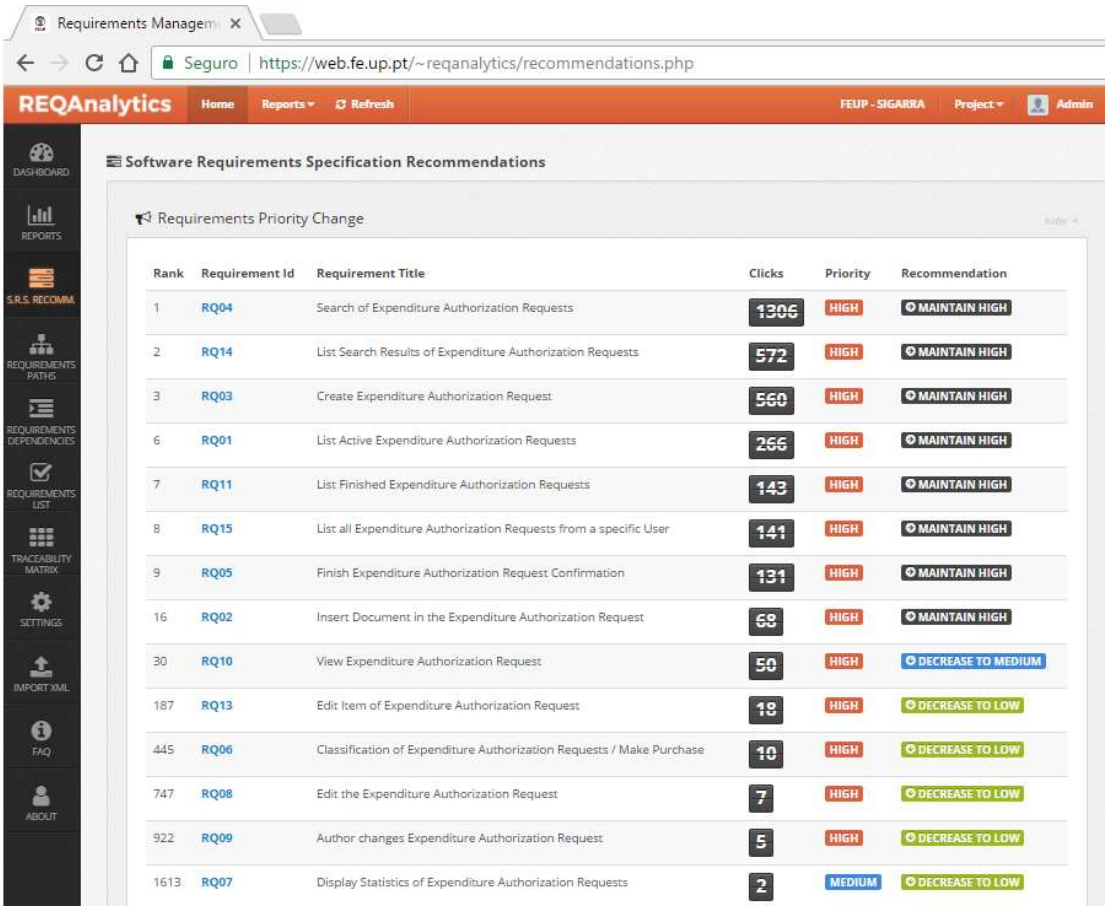


Figure 4: Example of recommendation to change priority.

The system also provides management reports, such as:

- **Requirement Details -** A detailed description of each requirement with all its properties and the information about the accesses to the corresponding functionality (that may be inferred by the accesses to the web pages and elements mapped to the requirement).

- **Dashboard** - A page showing the requirements already mapped; number of established mapping; charts indicating the percentage of the requirements by priority; and total number of recommendations generated by REQAnalytics.

- **Requirements Dependencies -** A visual graph that shows the dependencies detected by REQAnalytics for each of the requirements with previous and subsequent requirement. The nodes of this chart represent the functional requirements and the edges represent a dependency between two requirements.

- **Most used Requirements Navigation Paths -** Shows the most frequent paths taken by users along the web service, here the user visualizes functional requirements executed along those paths, and not the sequences of web pages visited, as in other tools.

- **Requirements Analytics (Statistics and Main Metrics)** - A list with the most accessed Requirements, Entry Requirements, Exit Requirements and Requirements Bounce Rate.

Mapping Tool

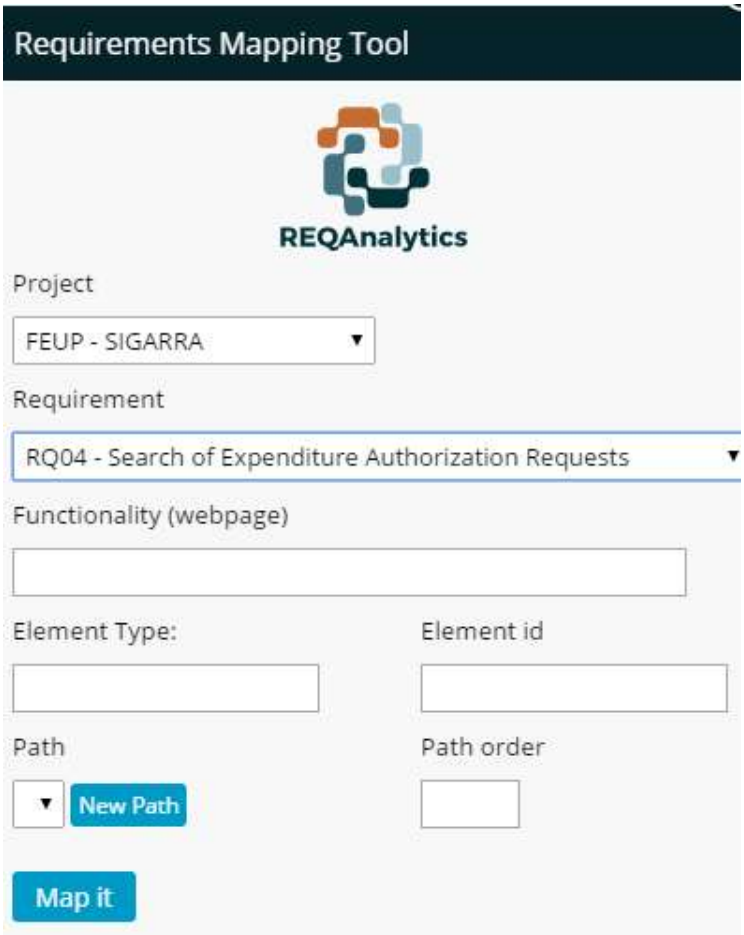


Figure 5: Mapping tool

REQAnalytics includes a mapping tool that is a bookmarklet, which runs on web browsers. The tool is used to create the mapping from a functional requirement to a web page. It can also correlate a sequence of different web pages, or elements, to a functional requirement. This

traceability tool has the ability of mapping a single functional requirement to multiple web pages, or elements, and a single web page to multiple functional requirements. Figure 5 shows the actual tool and its fields.

The user firstly selects a requirement in a combo box list with all the functional requirements imported from the XML document, and then clicks on the web page, or HTML element, that is related to the selected requirement. If the requirement is related to a sequence of web pages or HTML elements, the user will have to select all the web pages and elements. The final step is to confirm the mapping by clicking the button "Map it". The mapping tool then saves the created traceability link to the REQAnalytics' MySQL database. Figure 6 shows the workflow of the mapping tool.

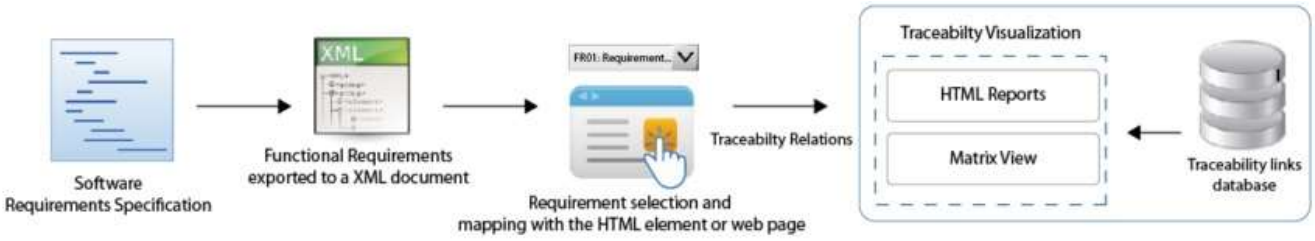


Figure 6: Mapping tool workflow

2.5 Open Web Analytics

Open Web Analytics (OWA) is a free and open-source software that is used to collect web usage data. It works along with a MySQL database. Its major features include the support for multiple websites, showing aggregated or individual statistics, tracking of all DOM elements clicked on each web page. It also has a set of tracking reports, namely beginning and ending of sessions, click-streams (clicks path), pages viewed, clicked elements, clicks tracking, click heat maps, entry and exit pages, among other features that are not necessary for this paper.

The key feature is the tracking of all DOM elements of the webpage; it allows us to know exactly what element was clicked, showing also its HTML identifier. It makes possible for the mapping tool to correlate a functional requirement with a single element of a web page (e.g. input fields) and not just with the URL of the web page. Since it is common to have more than one functionality in a single web page, this feature enables REQAnalytics to create a more detailed mapping of one requirement to the single elements inside the web page. Figure 7 shows the dashboard of the OWA tool.

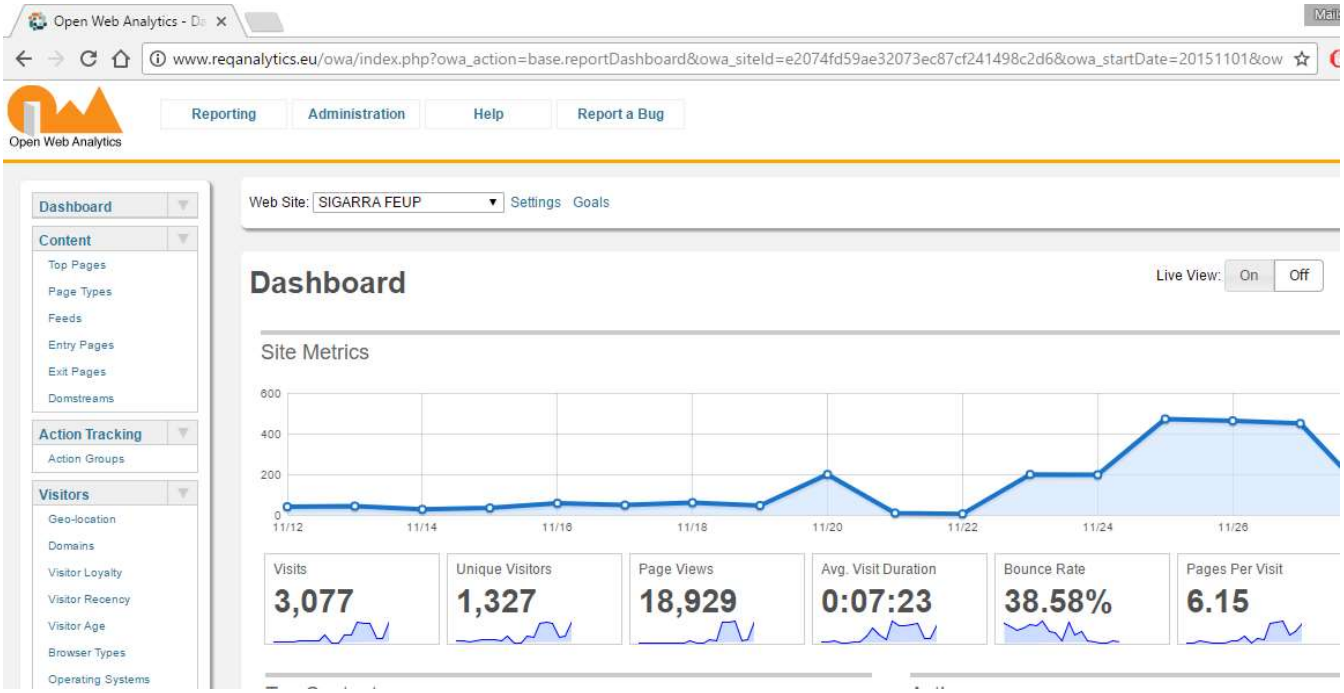


Figure 7: Open Web Analytics Dashboard.

OWA uses a MySQL database to store all the web usage data collected from the websites. Since REQAnalytics shares the same database, it is possible to execute SQL queries joining both OWA and REQAnalytics tables, directly in the database. Additionally, OWA has an extension that can be used to export data in JSON, XML, serialized PHP, and basic HTML.

To collect the data of web usage, OWA uses a tracking code that needs to be inserted in each analysed webpage source code. The tracking code is a JavaScript code that collects and sends data to the OWA database from each web page. An example of a tracking code is shown in Figure 8.

```
<script type="text/javascript">
var owa_cmds=owa_cmds || [];
owa_cmds.push(['setSiteId', 'a8e423432e9892a42124a31']);
owa_cmds.push(['trackPageView']);
mapping tool 79
owa_cmds.push(['trackClicks']);
owa_cmds.push(['trackDomStream']);
(function() {
    var _owa=document.createElement('script');
    _owa.type='text/javascript';
    _owa.async=true;
    owa_baseUrl.replace(/http:/,'https:') z:owa_baseUrl );
    _owa.src=owa_baseUrl + 'owa.tracker-combined-min.js';
    var _owa_s=document.getElementsByTagName('script')[0];
    _owa_s.parentNode.insertBefore(_owa, _owa_s);
}());
</script>
```

Figure 8: OWA JavaScript tracking code example.

2.6 Final Considerations

The automation of software quality for requirements documentation is an appealing idea, and the connection of requirements engineering with web usage mining techniques provides a powerful tool for Software Engineers. Automated approaches, when done right, are a more reliable source of quality information, therefore increasing the overall customer's satisfaction with the website, facilitating the work of the technical team, and improving the reliability of the software.

Chapter 3

REQAnaytics – Requirements Dependencies discovery

Having traceability information updated is useful to perform impact analysis when there are change requests and help to implement those changes without degrading the quality of the website. Impact Analysis is defined as ”identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change” [Bohner & Arnold, 1996]. By analysing the data about the usage of websites, it is possible to extract information about all sequences of functionalities executed on all sessions and with that identify new possible dependencies among requirements that were not identified previously.

This chapter firstly characterizes the REQAnalytics functionality for visualization of the requirements dependencies, which is the base for the Case Study on Chapter 4. The second part of the chapter explains the methodology applied in this dissertation, as well as the proposed extensions to REQAnalytics.

3.1 Activities performed

To better visualise the steps present in the proposed approach, we created an Activity Diagram, which is showed in Figure 9.

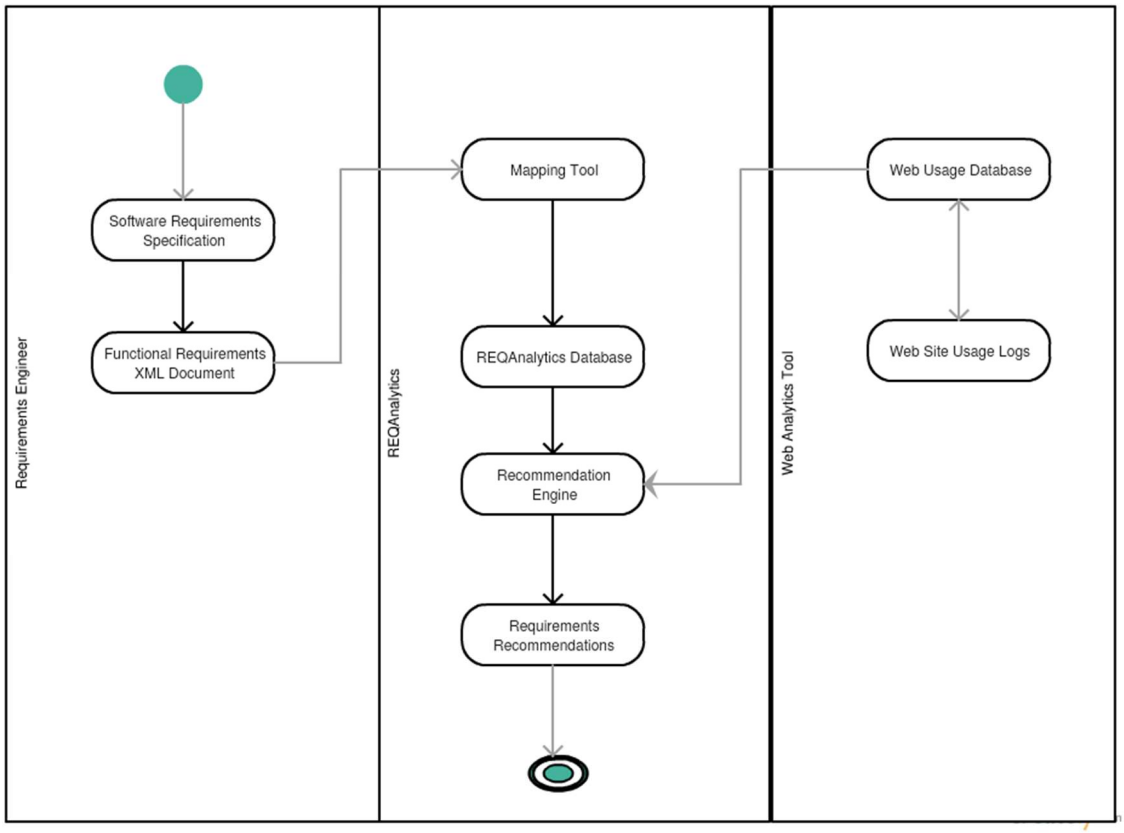


Figure 9: Activity diagram of performed approach.

As we can see in the Figure 9 above, first phase is where the Requirements Engineer collects all the functional requirements and transforms it into an XML document, which will be later imported to REQAnalytics.

Entry requirements

After importing the requirements XML document, the next step is to create the mapping between functional requirements and web page(s). With the information generated by the mapping tool, REQAnalytics is able to create a Traceability Matrix Report. This report is useful to check if the user has completed the mapping phase, where all requirements/functionalities of the website are mapped to the corresponding web page(s) or UI element(s). Figure 10 shows an

example of this report, the columns are the functional requirements and the lines are the web page elements.

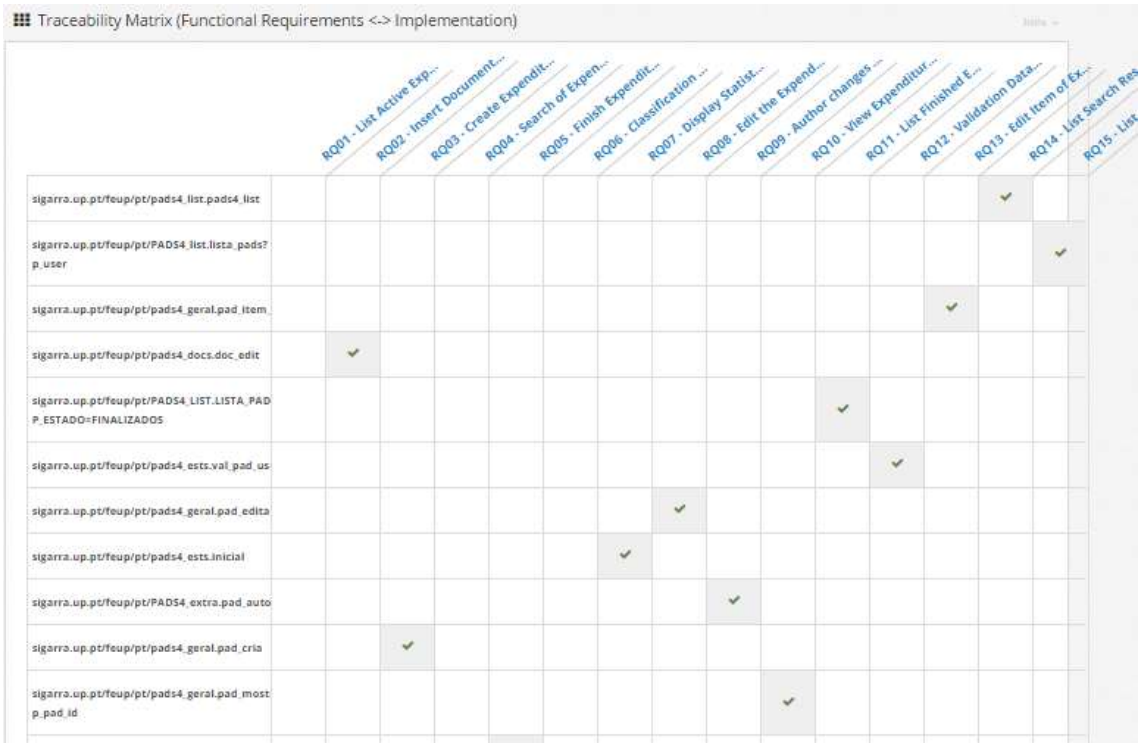


Figure 10: REQAnalytics Traceability Matrix Report

Requirements Details

With the functional requirements imported and the mapping stabilished, REQAnalytics then shows all the SRS details in a web page, containing also information about functional requirements that depends on the selected and the ones that are its dependants. Figure 11 shows an example of a requirement detail.

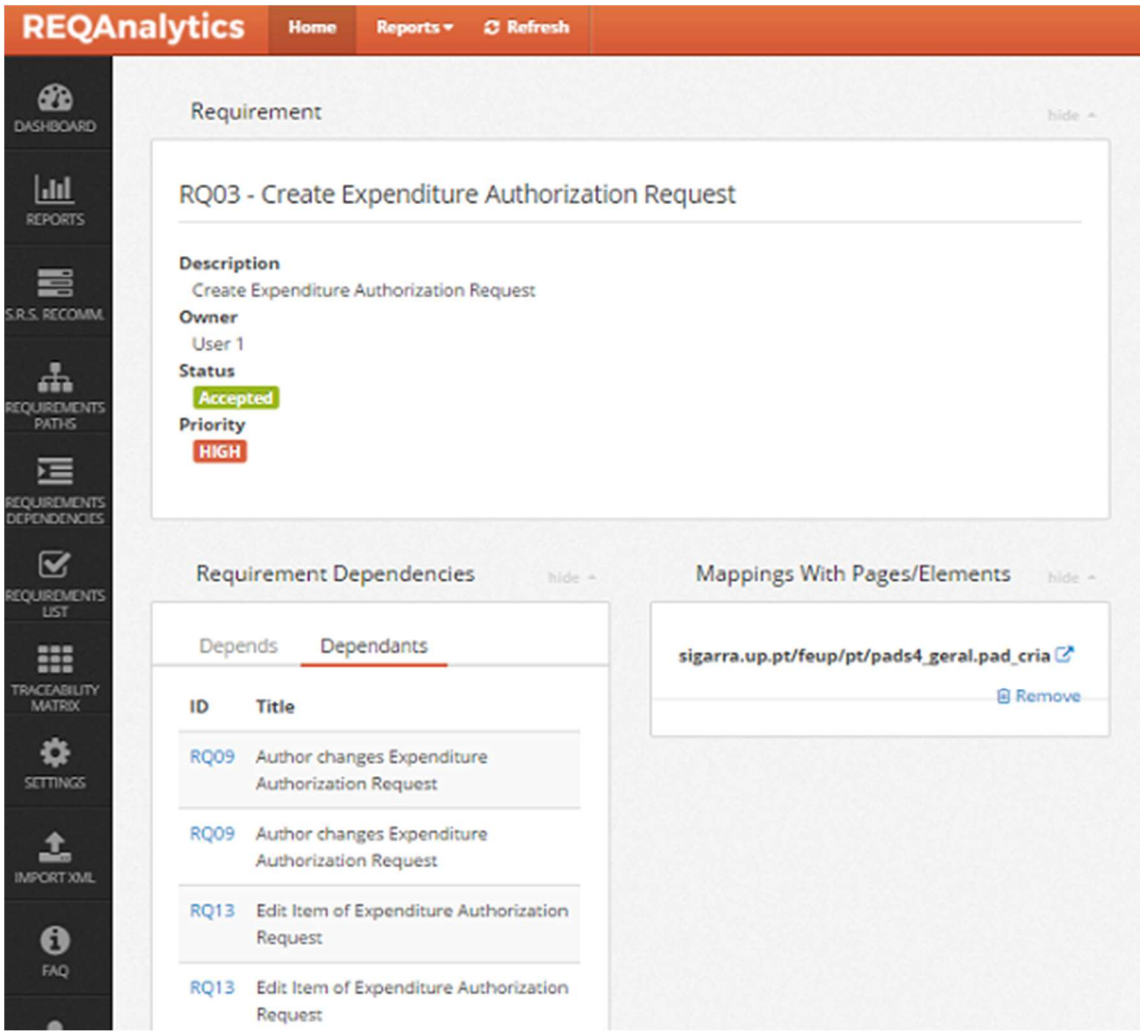


Figure 11: Requirements Details

3.2 Collection of Web Usage data

The goal of this phase is to collect the web usage data available from a web analytics tool, in our case the selected tool is Open Web Analytics (OWA), as referred before. It will collect several data like pages visited, DOM elements clicked, click paths, entry pages, exit pages and duration of session. This data will be stored in a support MySQL database for further analysis.

In this phase, REQAnalytics analyses the web usage data collected by the web analytics tool, linking it with the mapping information stored in the previous phase. In this analysis, possible improvements for the website are searched. The paths taken by users while visiting the website are identified for, among other improvements, show possible shorter paths, recommend workflow changes, describe which are the most and least used requirements, create new requirements not provided in the previous requirements specification, change the priority of the requirements and

detect navigation paths patterns. For the purpose of this research, we look specifically for possible new dependencies not declared in the original functional requirements XML document.

3.3 Graph Theory

Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph is an ordered pair G = (V, E) made up of a set of V vertices (or nodes, or points), which are connected with a set E of edges (or arrows, or lines). A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another. Figure 12 shows a drawing of a directed graph.

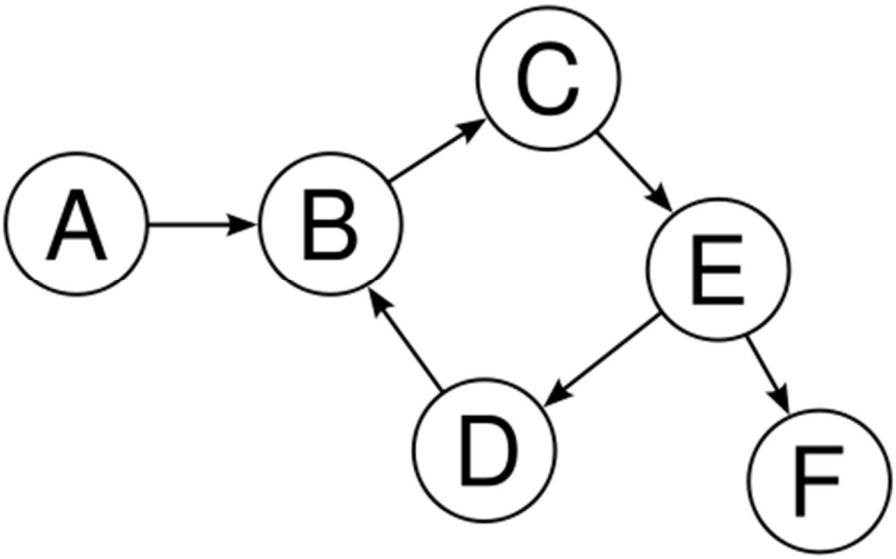


Figure 12: Example of a directed graph.

Directed Graph is a set of vertices (also called nodes or points) and a collection of directed edges (often-called arrows or directed lines) that each connects an ordered pair of vertices. A directed edge points from the first vertex in the pair to the second vertex in the pair. Thus, the first vertex has the lowest order in the pair.

In formal terms, a Directed Graph G is an ordered 4-tuple G: = (V, A, s, t) with

- V a set of vertices or nodes,
- A a set of edges or lines,
- ***s : A -> V***, assigning to each edge its source node,
- ***t : A -> V***, assigning to each edge its target node.

3.4 Requirements Dependencies Directed Graph

A Requirements Dependencies Directed Graph is a graph that shows the dependencies detected by REQAnalytics, each requirement connected with its previous and subsequent requirement. The nodes (circles) of the graph represent the functional requirements and the edges (arrows) represent a dependency between two requirements, as shown in Figure 13.

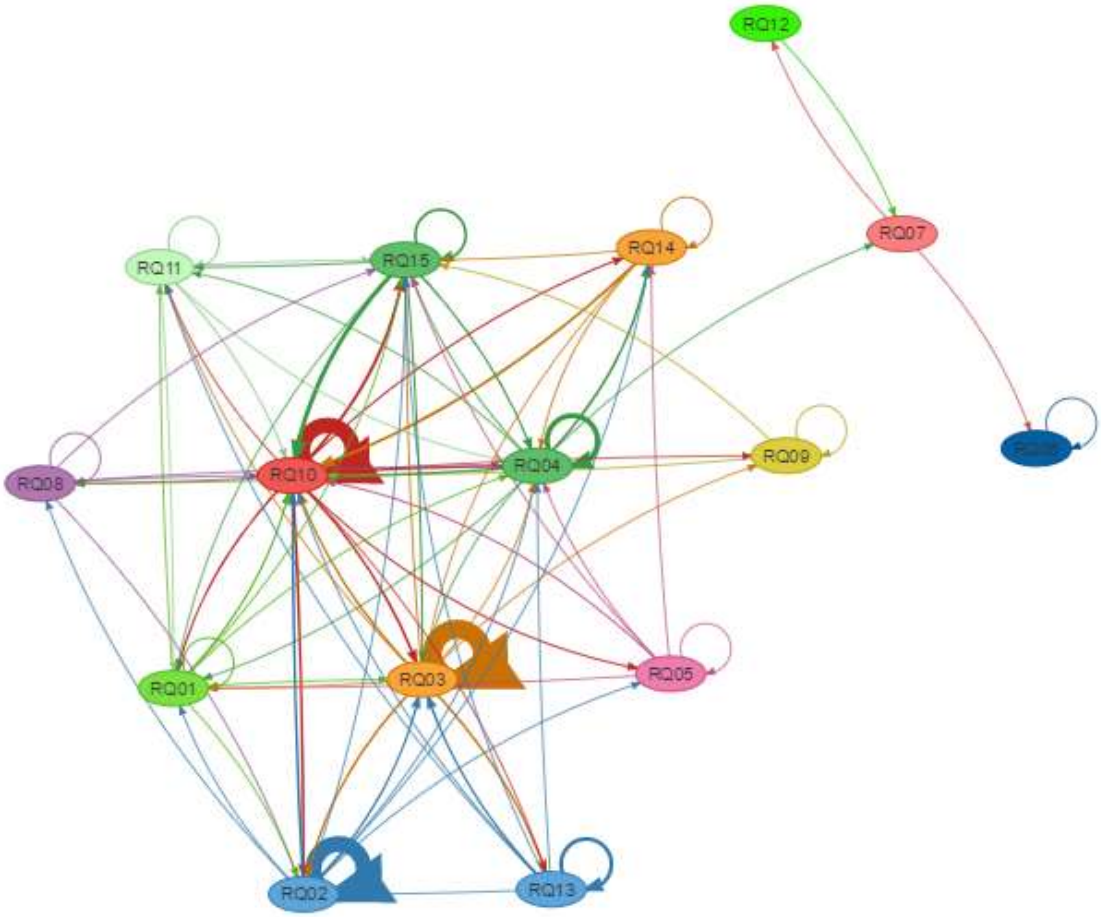


Figure 13: REQAnalytics Requirements Dependencies Directed Graph

Directed Graph analysis

This graph allows us to visualize the execution order of the requirements and all the navigations paths taken by the users within the website, from the perspective of the functionalities (i.e., requirements) performed. REQAnalytics automatically generates this graph based on the information of web usage collected by OWA, correlated with the mapping information of the requirements. For the purpose of this paper, we analyse the data generated by users' clickstream, what is the record of what parts of the screen were clicked by the user and in what order.

With this graph, it is also possible to detect what are the feasible dependencies between the requirements and what are the paths of functionalities most used on the website. The thickness of the line indicates the number of clicks between those two requirements. The more clicks exist between two requirements, the thicker the line is. Figure 13 shows a screenshot of the directed graph automatically generated in the case study.

3.5 REQAnalytics – Extensions developed

Throughout an evaluation of the existing tools available in REQAnalytics, it was possible to detect two improvement points, which are:

1) Import an XML file containing the functional requirements list with its attributes and dependencies;

2) Utilize the imported functional requirements to create a traceability matrix, correlate them with the web usage data, and then recommend new dependencies that may be created.

First point will be dealt with by applying a well-defined market standard, the researcher found that the ReqIF [Group, 2016] standard is the best option to achieve a high-quality development, due to its big community, available support, and constant updates. The standard is defined on Section 3.2.1.

The second point requires the application of Graph Theory [Bindner et al., 2014], where the Tarjan's Algorithm is used to calculate bridges between graph nodes, these nodes are collected from the REQAnalytics Requirements Dependencies Directed Graph functionality.

All the web usage data analysed in the Case Study (Chapter 4) were already available from previous work [J. E. Garcia & Paiva, 2016a], what differs this report is the creation of an XML file in the ReqIF format and the context in which the information is used as source to the Tarjan's Bridge Finder Algorithm [Italiano, Laura, & Santaroni, 2012].

3.5.1 The ReqIF standard

The Requirements Interchange Format (ReqIF) standard is used to exchange requirements, along with its associated metadata, between software tools from different vendors. It is developed and maintained by the Object Management Group (OMG), same group responsible for the UML standard. The proposed extension is called 'Import XML' inside the REQAnalytics system; there

it is possible to import the ReqIF XML file (with .reqif extension) into the dependencies table of REQAnalytics (req_dependency).

The ReqIF XML structure consists in a metadata of requirements information, which means that it is not a hard coded standard where one can easily relate its original attributes to a destination field. Each vendor can export a .reqif file that complies with the standard but has their own names and conventions for the same fields.

The ReqIF file is an XML containing a root element called REQ-IF, with information about the data types and the requirements it has, and its specification objects (element SpecObject), which contains the user-defined attributes. Each attribute has its data type, which include the basic types (String, Bool, Int, Float), and XHTML, which can be used to exchange formatted text and images, the file also contains information about the relationships between objects, which are called SpecRelations.

Afterwards, the functional requirements were exported to a ReqIF file, in this document, all the functional requirements used in this case study are described, and each one has its own attributes and dependencies, the latter only when such dependency exists. Figure 14 shows a code snippet of the ReqIF XML document with an example of a dependency.

```
<SPEC-RELATIONS>
  <SPEC-RELATION IDENTIFIER="ADB3C6E4-8014-4167-9D21-A8E13D98C6CA" LAST-CHANGE="2017-01-01T10:24:18.000+01:00">
    <TARGET>
      <SPEC-OBJECT-REF>RQ07</SPEC-OBJECT-REF>
    </TARGET>
    <SOURCE>
      <SPEC-OBJECT-REF>RQ06</SPEC-OBJECT-REF>
    </SOURCE>
    <TYPE>
      <SPEC-RELATION-TYPE-REF>depends</SPEC-RELATION-TYPE-REF>
    </TYPE>
  </SPEC-RELATION>
</SPEC-RELATIONS>
```

Figure 14: Snippet of the ReqIF document with an example of a dependency

After importing the information of the ReqIF file, REQAnalytics builds a traceability matrix of the relationships contained on the SpecRelations object. These relationships are based on the type “depends on”, where a requirement specification A (target) depends on the requirements specification B (source). Figure 15 shows the extension to import the functional requirements.

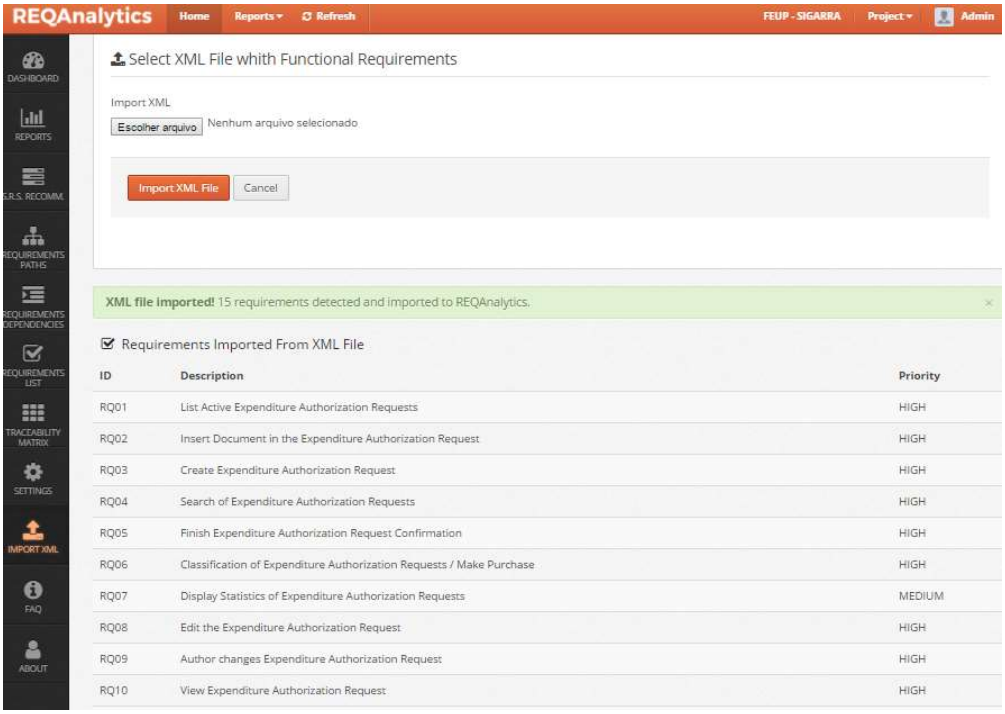


Figure 15: Extension to import functional requirements

Supporting Tools

As the ReqIF is an open standard, several tools in the market have support to the format. Commercial tools like IBM Rational DOORS, Polarion, NoMagic MagicDraw, and ReqIF Server from enso, among others, all of them have functionalities to import, export and edit ReqIF files. In the open source community, the most active implementation is the Eclipse Requirements Modelling Framework [RMF], maintained by the Eclipse Foundation and, of course, based on Eclipse. Eclipse is an open source platform that has an integrated development environment (IDE) created for building Java applications, but actually can do much more than that.

The RMF tool includes a user interface called ProR, which is a stand-alone ReqIF editor, and is free for use. A version of ProR is distributed along the formalmind ReqIF Studio, available at http://formalmind.com/tools/studio/, we used this version for the creation of the functional requirements present in the Case Study on Chapter 4. Figure 16 shows the ReqIF Studio’s IDE.

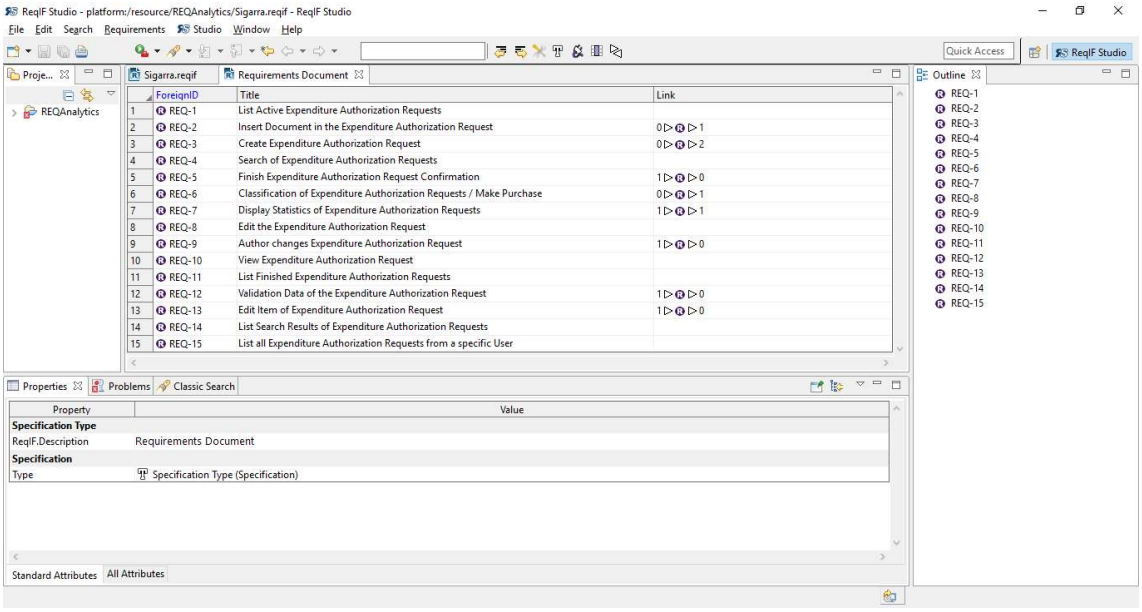


Figure 16: ReqIF Studio IDE.

The scenario on where the ReqIF standard is based on is one where constant exchanges of requirements specification files are needed, in projects where the teams are geographically distant or different partners working on different parts of the specification. Figure 17 shows this scenario.

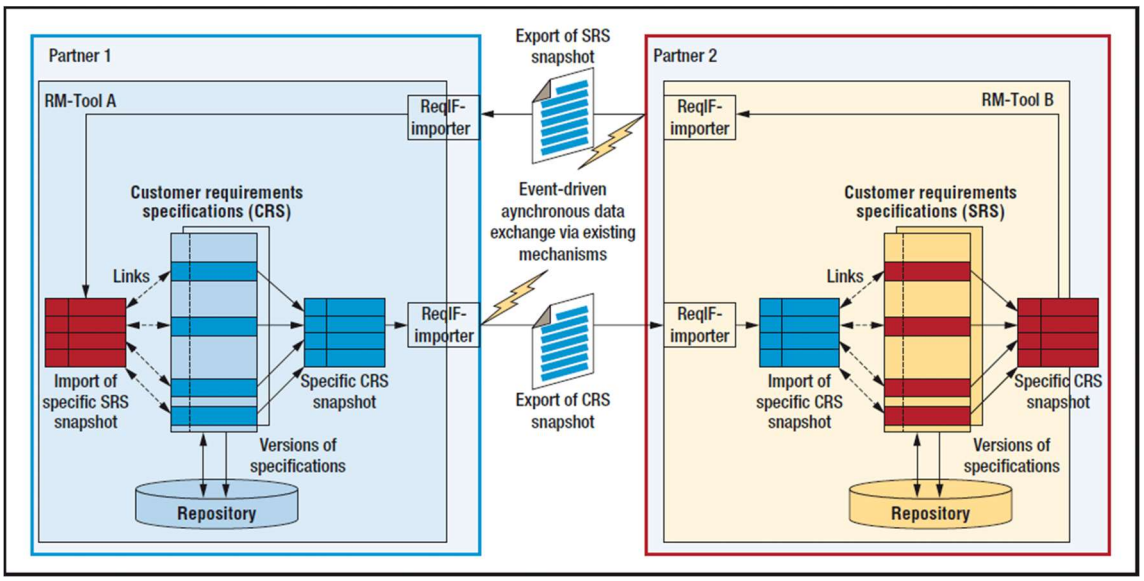


Figure 17: ReqIF exchange scenario.

3.5.2 Discovery of new dependencies

This extension proposes a new feature for REQAnalytics, on which the clickstream data of the website is analysed and a traceability matrix with suggestions of new requirements dependencies is generated. With the clickstream data, REQAnalytics knows the order in which the users executed the functionalities. Figure 18 shows the Requirements Dependencies Directed Graph of the functional requirements from the Case study.

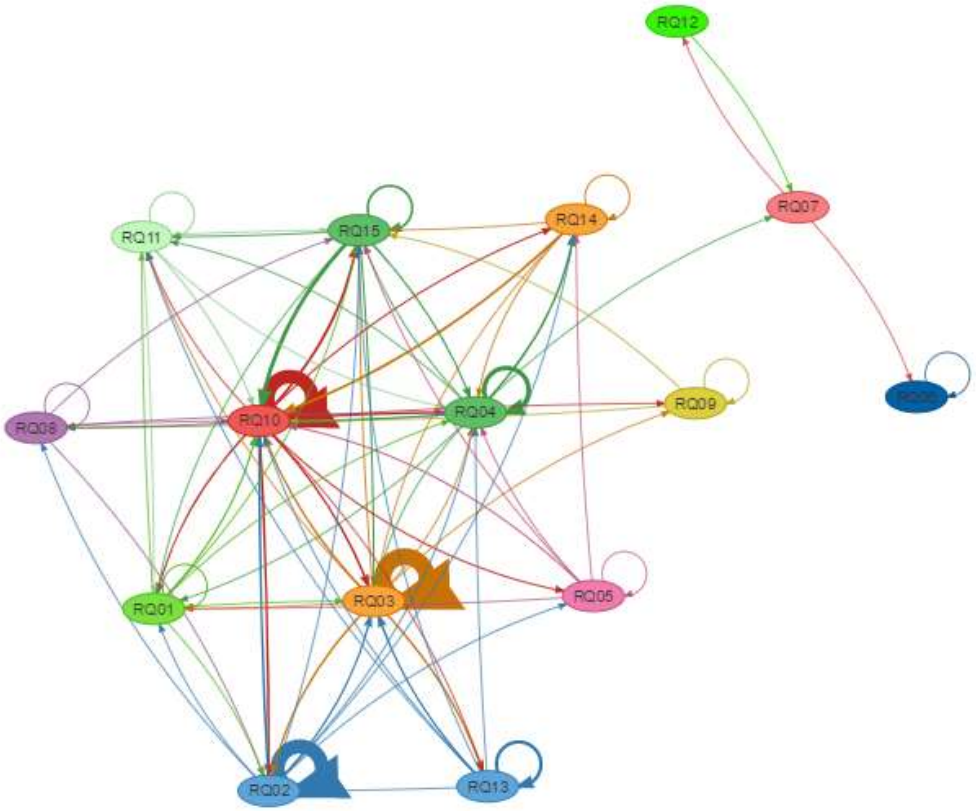


Figure 18: Requirements Dependencies Directed Graph from the Case Study.

After knowing the execution order, it is then possible to track back the dependencies imported from the ReqIF file and check if there are new dependencies that are not documented. For instance, if there is a functionality F2 that is accessed always after functionality F1, it may suggest that there is a dependency from F2 to F1.

To detect the new dependency we use an algorithm to try to find bridges between the edges of the graph. Bridge is an edge between two nodes, *v* and *t*, that after being removed there is no path left between *v* and *t*. An edge is a bridge if its removal increases the number of strongly

connected components [Italiano et al., 2012], if a bridge is found we can assume it as a dependency.

The first algorithm for finding bridges in a graph was described by Robert Tarjan in 1974 [Tarjan, 1974]. Generically, what the algorithm does is:

1 For every edge (u, v), do following

a Remove (u, v) from graph

b See if the graph remains connected (We can either use BFS or DFS)

c Add (u, v) back to the graph.

Time complexity of above method is O(E*(V+E)) for a graph represented as adjacency list. However, there are other less complex algorithms that we may consider implementing in the future.

```php
<?php
    function bridge($G_local){
        for ($v = 0; $v < count($G_local); ++$v){
            $visited[$v] = False;
            $disc[$v] = -1;
            $low[$v] = -1;
            $parent[$v] = -1;
        }
        for ($i = 0; $i < count($G_local); ++$i){
            if ($visited[$i] == False){
                bridgeUtil($i, $visited, $parent, $low, $disc);
            }
        }
    }

    function bridgeUtil($u, $visited, $parent, $low, $disc){
        foreach($G[$u] as $v) {
            if ($visited[$v] == False){
                $parent[$v] = $u;
                $children = $children+1;
                bridgeUtil($v, $visited, $parent, $low, $disc);

                $low[$u] = min($low[$u], $low[$v]);
                if ($low[$v] > $disc[$u]){
                    echo "new bridge:".$u."-".$v;
                }
            }
            elseif($v !== $parent[$u]){
                $low[$u] = min($low[$u], $disc[$v]);
            }
        }
    }
?>
```

.

Figure 19: Pseudocode of the bridge finder algorithm in PHP

Figure 19 shows the pseudocode of the extension, written in PHP. If the algorithm detects a bridge, the extension checks if the bridge, which is also a dependency, is contained in the original XML functional requirements file and, in case it is missing, recommends adding the identified dependency to the original functional requirements. This new recommendation is presented in Chapter 4 with a Case Study.

3.6 Discussion

In this Chapter, we presented the activities performed to achieve the goals defined in the Case Study, from the collection of the entry functional requirements, going through the existing functionalities of the REQAnalytics recommender system and proposing extensions to it. The extensions consist of a routine to import a functional requirements XML document in a defined market standard (ReqIF), and a Graph Theory algorithm to analyse the Requirements Dependencies Directed Graph looking for new, undetected requirements dependencies, which mat help the software and requirements engineers in the task of assessing the impact of changes request in websites. In the next Section, a Case Study is presented to validate all the method hereby proposed.

Chapter 4

Case Study

This chapter shows how the proposed approach, described in chapter 3, is implemented, the achieved results and a final discussion. This Case Study aims to validate the proposed approach and the extensions developed to the REQAnalytics recommender system. A functional demo of REQAnalytics system and the extensions developed in this project is available at https://web.fe.up.pt/~reqanalytics.

The Case Study here developed followed the method described by Robert Yin, in his book Case study research: Design and methods [Yin, 1984]. The method consists of a case study process containing six stages for qualitative research. Figure 20 shows the six stages.

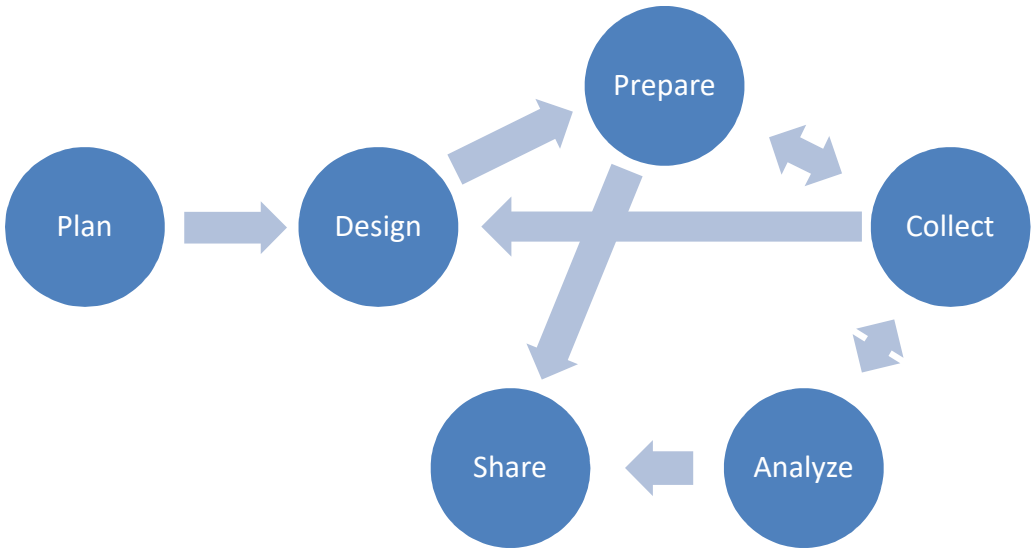


Figure 20: Robert Yin's Case Study process.

Step one – Plan: Identify the research questions and idealize the case study;

Step two – Design: Define what data will be collected, define quality criteria and quality procedures, and finally define the interpretation criteria for the findings;

Step three – Prepare: Define procedures for data collection and identify participant teams;

Step four – Collect: Gathering of the previously defined data, this phase is the actual case study realization.

Step five – Analyze: Is the step whre the collected data is examined, categorized, and tabulated to reach empirical conclusions;

Step six – Share: Report of the case study with findings and results.

The case study provides an analysis of the Web usage data, mapped to the functional requirements of a website, and its requirements dependencies, so that new requirements dependencies can be detected.

4.1 SIGARRA – FEUP

This section presents the results of the application of REQAnalytics in the web information system SIGARRA from the Faculty of Engineering of University of Porto Website. Due to the complexity of SIGARRA, it is analysed only one module called "Expenditure Authorization Requests Management". This is a private module of the web information system, and is used to support some of its business core functionalities. Figure 21 shows a screenshot of the module.

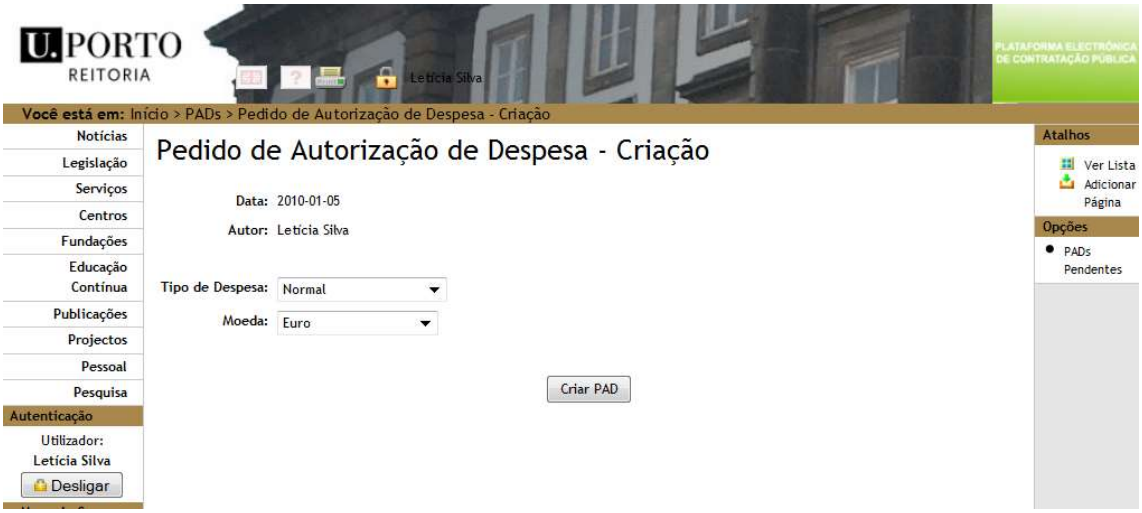


Figure 21: Main screen of the Expenditure Authorization module.

SIGARRA is the acronym in Portuguese for “Information System for Aggregate Resource Management and Academic Records”; it was developed at the University of Porto. In 1992 the Central Office team created GAUP – Gestão de Alunos da UP (Management of Students from the UP), which became of general use to all the UP faculties, with the purpose of managing the students’ records in their respective administration offices. After 1996, the project of an integrated information academic system, with web interface, called SiFEUP, was launched at FEUP. This System moved towards integration with GAUP, having a profound revision in 2001.

After 2003, the SIGARRA, an integrated system on the Web, was installed in the majority of the UP faculties. From that moment on, its availability was also admitted for other Higher Education Institutions.

The SIGARRA's Project of University of Porto was officially created in May 13th 2011.

4.1.1 Goals

The following goals were set for this case study:

- Successfully import requirements in the ReqIF format;
- Identify and recommend the creation of new dependencies from the web usage data;

4.1.2 Data Collection

As previously described in Chapter 3, the first phase for the REQAnalytics system is to set the functional requirements of the website that will be used by the analysis performed by the recommender system engine.

The functional requirements used in this Case Study were defined by the technical development team at FEUP – CICA (Centro de Informática Prof. Correia de Araújo), who are the developers of the main SIGARRA system and all its secondary modules. Table 1 describes the functional requirements collected along with the associated dependency to other functional requirements. Each functional requirement has its unique identifier (column ID), Description, Priority Level, and the last column has the ID for the functional requirement it is dependant.

A relationship of the type “depends on” means that the requirement specification A depends on the requirement specification B, a change or update in A may affect the requirement specification contained in B. In Table 1, we can see that requirement “RQ02 - Insert Document

in the Expenditure Authorization Request" has the requirement "RQ05 - Finish Expenditure Authorization Request Confirmation" as its dependant. By simply reading the functional requirements list, we can presume that is not possible to reach or perform requirement RQ05 before requirement RQ02 is initiated and/or finished. We can also say that a change in requirement RQ02 may affect the requirement RQ05 and it is necessary to analyse both together when such change occurs.

The data collected showed five dependencies among the functional requirements, all of them were defined by the team of specialists at FEUP – CICA, who are responsible for the development of SIGARRA website.

Table 1 Functional Requirements analysed in the Case Study.

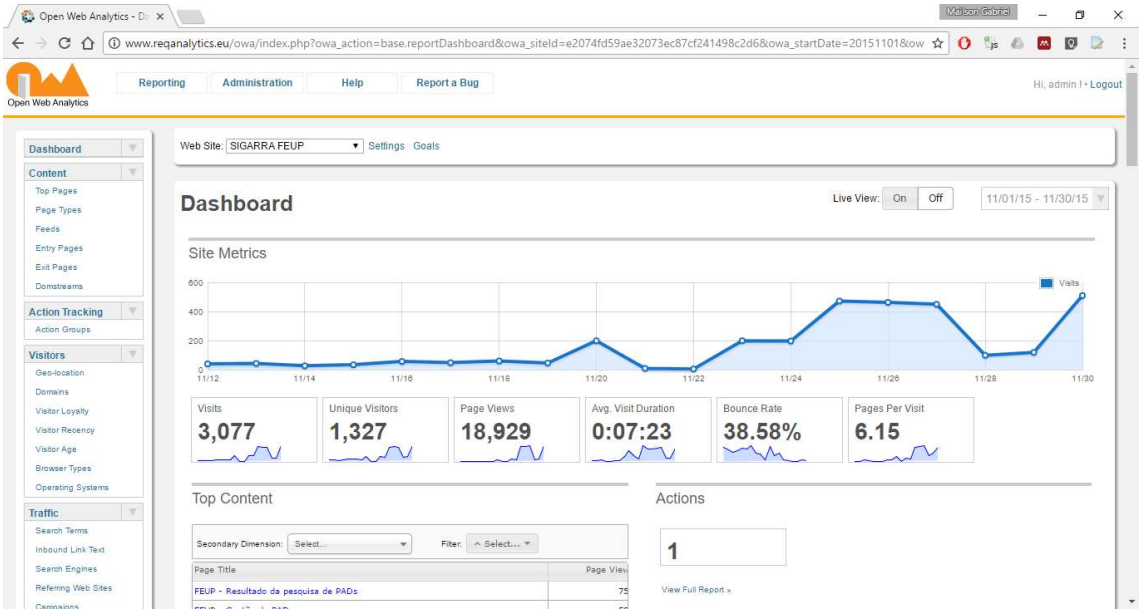
ID	Description	Priority	Depends
RQ01	List Active Expenditure Authorization Requests	High	
RQ02	Insert Document in the Expenditure Authorization Request	High	
RQ03	Create Expenditure Authorization Request	High	
RQ04	Search of Expenditure Authorization Requests	High	
RQ05	Finish Expenditure Authorization Request Confirmation	High	RQ02
RQ06	Classification of Expenditure Authorization Requests / Make Purchase	High	
RQ07	Display Statistics of Expenditure Authorization Requests	Medium	RQ06
RQ08	Edit the Expenditure Authorization Request	High	
RQ09	Author changes Expenditure Authorization Request	High	RQ03
RQ10	View Expenditure Authorization Request	High	
RQ11	List Finished Expenditure Authorization Requests	High	
RQ12	Validate Data of the Expenditure Authorization Request	High	RQ07
RQ13	Edit Authorization Request	High	RQ03

ID	Description	Priority	Depends
RQ14	List Search Results of Expenditure Authorization Requests	High	
RQ15	List all Expenditure Authorization Requests from a specific User	High	

Experiment

The functional requirements described in Table 1 were exported to an XML document in the REQIF format, which is described in Section 3.2.1. Afterwards, the XML document was imported to the REQAnalytics database using the import routine also described in Section 3.2.1. Then the mapping between each requirement and the web pages and elements that implements it was established by a mapping tool. This mapping was made using the web based mapping tool included in REQAnalytics system, described in Section 2.4.

The web usage data for this case study were collected using a Web Analytics tool (OWA). The time period considered was 1 month (November 2015) before the recommender system analysis was carried out. During the time period 3.077 visits with 1.327 unique user sessions where registered. Each user session consists in a log of clicks generated by the SIGARRA website users. Figure 22 shows the registered quantity of visits and sessions.



4.1.3 Results

With the information of the traceability links and the functional requirements stored in the database, REQAnalytics system is able to generate a traceability matrix between functional requirements. Figure 23 shows the traceability matrix generated with the data imported in the previous section.

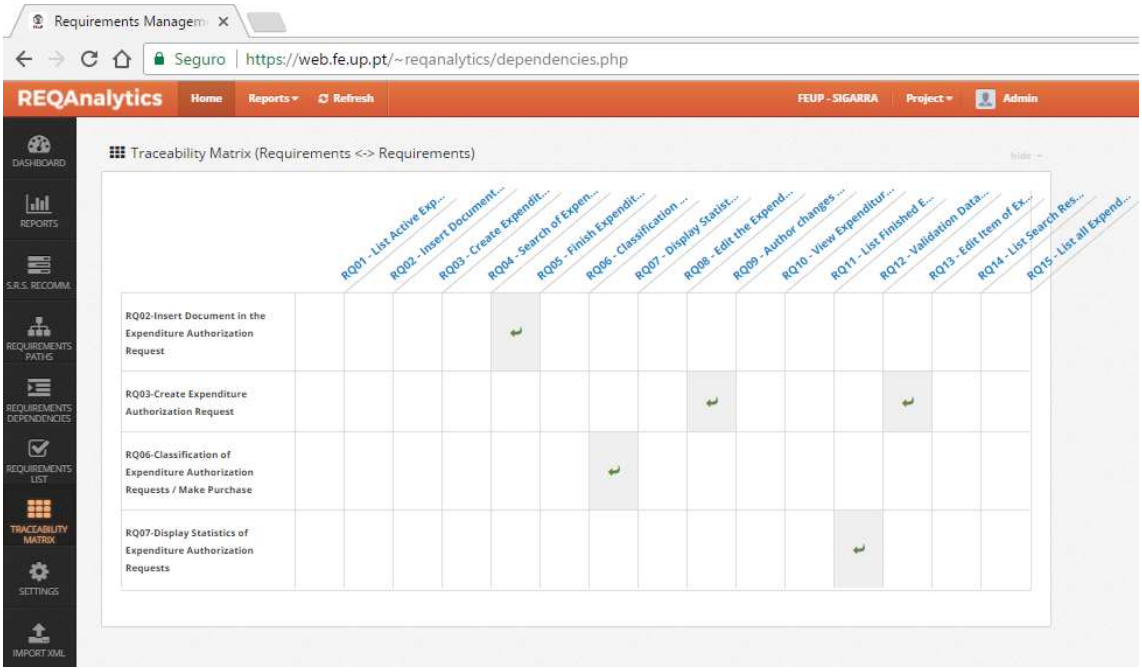
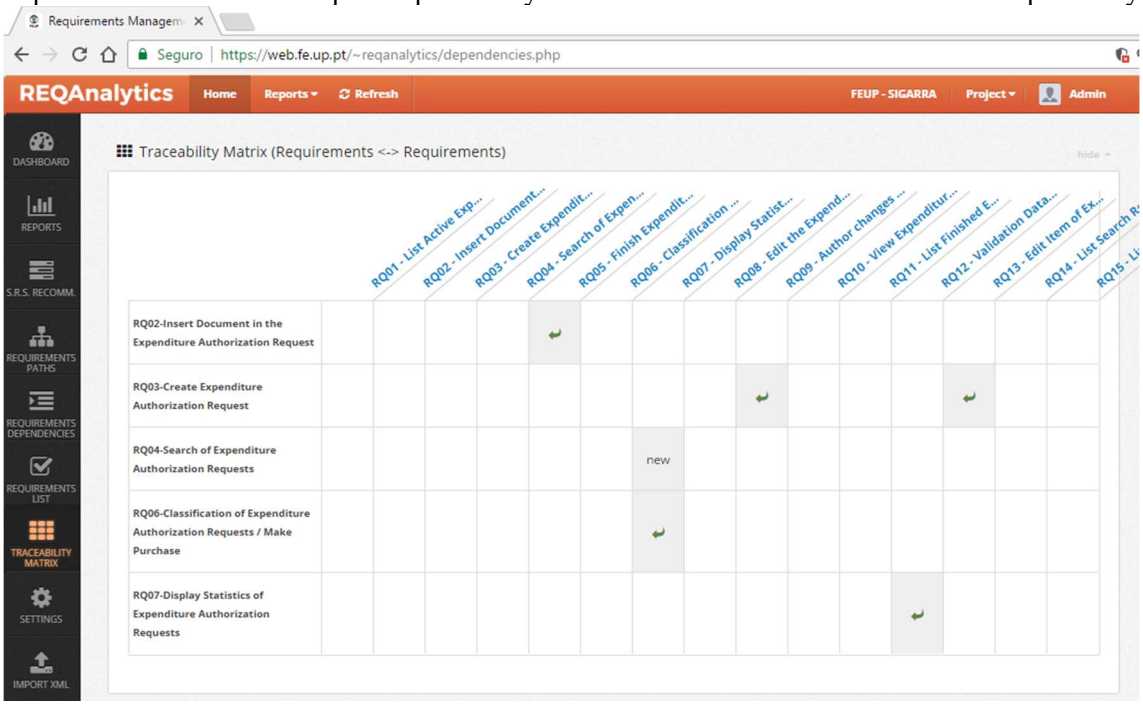


Figure 25: Dependencies extracted from the original XML requirements document.

Figure 26: New dependency detected from web usage logs.Figure 27: Dependencies extracted from the original XML requirements document.

The extension described in Section 3.5.2 used the data from REQAnalytics Requirements Dependencies Directed Graph, and transformed it into an adjacency matrix. With this adjacency matrix as source, the bridge finder algorithm could detect a bridge between requirements RQ07 and RQ04, which was not present in the initial functional requirements XML document. REQAnalytics has now the ability to recommend the addition of new requirements dependencies based on the web usage data. The extension generates a new traceability matrix with the recommendation. This new traceability matrix shows all the functional requirements

dependencies that where imported previously from the XML document and the new dependency



detected by the extension. Figure 24 shows a screenshot of this new traceability matrix, where the column 'new' suggests a new dependency between functional requirements RQ04 and RQ07.

4.2 Discussion

With the Case Study, it is possible to validate the execution of both goals proposed in Section 4.1.1, with an analysis of the web usage information of the website SIGARRA.

First goal was to automate the routine of importing functional requirements to the REQAnalytics database, hence all requirements dependencies. This automation is needed because a manual input would be very time consuming for the software engineers and/or for the system analysts who use the system. Since after the recommendations it is necessary to update the source of the functional requirements and import its updated version, adopting this scenario helps REQAnalytics to have good quality and updated data without harassment.

Second, and most important, is the demonstration of how the concepts of Web Usage Mining, Requirements Management, and Graph Theory can be combined to develop a solution for Requirements Engineers to use in favour of an improvement in quality and reliability of a

Figure 28: New dependency detected from web usage logs.

website. These improvements come as recommendations that are not made by any other available tool in the market right now, or even other studies concerning Requirements Engineering.

The results suggest that recommender systems for software engineering, combined with Web Usage analysis, can be used to help the requirements maintenance during the lifecycle of a website, where requirements are in constant change and evolution, thus helping to improve the overall quality of the website.

The next chapter discusses the research done in this work, summarizing the contributions made and indicates directions for future work.

Chapter 5

Conclusion

This research work presents REQAnalytics, a recommender system that maps the functional requirements of a website to the features that implements them, relating this information with the web usage data so that it may generate recommendations to the requirements specification. Those recommendation reports may help the requirements maintenance and increase the overall quality of the software requirements specification and the whole website.

The current available Web analytics tools do not produce the type of analysis performed by REQAnalytics, since those tools only produce reports with navigation statistics. Moreover, current tools commonly only have traceability information between requirements and software test cases, and the majority of these tools is unable to automatically generate and maintain traceability relationships.

An evolution to the original REQAnalytics set of tools created the conditions to automate, and improve, the import of Software Requirements Specifications (SRS) along with its traceability information, via a well-defined market standard, which is an XML file in the .REQIF standard. In addition, we created a method for validation of the imported requirements, more specifically to its dependencies. After the SRS's are imported, REQAnalytics creates a traceability matrix containing the dependencies among the requirements, these dependencies still only the imported ones. Using information from Open Web Analytics (OWA), the system creates a Requirements Dependencies Directed Graph that shows the paths that users from the website took when navigating, analysing this graph is possible to identify dependencies among requirements and, when not previously defined, recommend the creation of new dependencies in the original SRS documents.

5.1 Contributions

The results of the Case Study demonstrate that the analysis of the website usage data can provide recommendations that allow the website to fulfil the expectations of its stakeholders. Such recommendations may be a great addition to the existing software development process of a website. Furthermore, when comparing this approach with existing web analytics and related tools, REQAnalytics presents more readable and understandable reports for stakeholders.

The results show that this approach may be useful to detect dependencies among requirements that were not documented initially. It was also possible to create an extension to implement an automated import of the functional requirements, via XMLs files created in the ReqIF standard.

Another advantage of this approach is that it also supports the task of requirements management, which contributes to the overall quality of the web service itself. The results also indicate that the recommendation provided by REQAnalytics performed better that the random case. Nevertheless, additional work is necessary to compare these results to those obtained using other types of recommendations.

5.2 Future Work

Improvements to the REQAnalytics recommender system can be made to use other Web Mining techniques to analyse and process the Web usage data. There are nowadays new methods and techniques applied to Web Usage Mining that may help to enhance the recommendations generated by REQAnalytics.

Now the only source used for the web usage data is the Open Web Analytics (OWA) tool, REQAnalytics directly access the OWA database to collect the information needed. We also aim to extend REQAnalytics in order to be possible to work with different web analytics tools since each of these tools can provide different kinds of data and metrics and because each website may use a different web analytics tool. The creation of a middle layer could enable the use of other sources, such as webserver log files. However, the implementation of such feature would require a great effort to change the already existing queries and algorithms.

We may consider implementing in the future other algorithms with less complexity, than the Trajan´s algorithm.

The scope of REQAnalytics can be extended to the analysis of applications usage on different platforms, such as tablets and mobile applications. With the popularity of mobile platforms, it may be a promising research.

Bibliography

Adeniyi, D. A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method Production and hosting by Elsevier. *Applied Computing and Informatics*, *12*, 90–108. https://doi.org/10.1016/j.aci.2014.10.001

Akerkar, R., Bădică, C., & Burdescu, D. D. (2012). Desiderata for research in web intelligence, mining and semantics. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics - WIMS '12* (p. 1). New York, New York, USA: ACM Press. https://doi.org/10.1145/2254129.2254131

Al-Rawas, A., & Easterbrook, S. (1996). Communication Problems in Requirements Engineering: A Field Study. *Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London*, (February), 1–12. https://doi.org/10.1.1.30.4408

Babar, M. I., Ramzan, M., & Ghayyur, S. A. K. (2011). Challenges and future trends in software requirements prioritization. In *International Conference on Computer Networks and Information Technology* (pp. 319–324). IEEE. https://doi.org/10.1109/ICCNIT.2011.6020888

Barbir, A., Hobbs, C., Bertino, E., Hirsch, F., & Martino, L. (2007). Challenges of Testing Web Services and Security in SOA Implementations. In *Test and Analysis of Web Services* (pp. 395–440). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72912-9_14

Beecham, S., Hall, T., Britton, C., Cottee, M., & Rainer, A. (2005). Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software*, *76*(3), 251–275. https://doi.org/10.1016/j.jss.2004.06.004

Beil, D. (2009). Supplier Selection Based on a Neural Network Model Using Genetic Algorithm. *IEEE Transactions on Neural Networks*, *20*(9), 1504–1519. https://doi.org/10.1109/TNN.2009.2027321

Bindner, D., Erickson, M., Blanchet-Sadri, F., Bóna, M., Brualdi, R. A., Chao, K.-M., … Yin, J. (2014). *Handbook on Graph Theory*. Retrieved from http://www.crcnetbase.com/doi/pdf/10.1201/b16132-1

Bohner, S. A., & Arnold, R. S. (1996). *Software Change Impact Analysis*. *Aviation*. IEEE Computer Society Press. Retrieved from http://portal.acm.org/citation.cfm?id=525066

Bubenko, J. a. . J. (1995). Challenges in requirements engineering. *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, 160–162. https://doi.org/10.1109/ISRE.1995.512557

Cancian, M. H. (2013). UM MODELO DE CAPACIDADE E MATURIDADE PARA

Candan, K. S., Li, W.-S. L. W.-S., Phan, T., & Zhou, M. Z. M. (2009). Frontiers in Information and Software as Services. *2009 IEEE 25th International Conference on Data Engineering*, 1762–1769. https://doi.org/10.1109/ICDE.2009.168

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Dag, J. N. och. (2001). An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, 84–92. https://doi.org/10.1109/ISRE.2001.948547

Cho, Y. H., Kim, J. K., & Kim, S. H. (2002). A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, *23*(3), 329–342. https://doi.org/10.1016/S0957-4174(02)00052-0

Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, *29*(9), 796–810. https://doi.org/10.1109/TSE.2003.1232285

Cooley, R., Mobasher, B., & Srivastava, J. (1999). Integrating Web Usage and Content Mining for More Effective Personalization. *Knowledge and Information Systems*, *1*(1), 5–32. https://doi.org/10.1007/BF03325089

Damian, D. E. (2007). Requirements Engineering challenges in multi-site software development Didar Zowghi Departments of Software Engineering University of Technology , Sydney. *Requirements Engineering*, *8*(3), 149–160. https://doi.org/10.1007/s00766-003-0173-1

De Lucia, A., & Qusef, A. (2010). Requirements engineering in agile software development. In *Journal of Emerging Technologies in Web Intelligence* (Vol. 2, pp. 212–220). https://doi.org/10.4304/jetwi.2.3.212-220

Dömges, R., & Pohl, K. (1998). Adapting traceability environments to project-specific needs. *Communications of the ACM*, *41*(12), 54–62. https://doi.org/10.1145/290133.290149

Facca, F. M., & Lanzi, P. L. (2004). Mining interesting knowledge from weblogs: a survey. https://doi.org/10.1016/j.datak.2004.08.001

Garcia, A., & Paiva, A. C. R. (2014). SaaS Usage Information for Requirements Maintenance, 323–330. https://doi.org/10.5220/0004898403230330

Garcia, J. E., & Paiva, A. C. R. (2016a). A Requirements-to-Implementation Mapping Tool for Requirements Traceability. *Journal of Software*, *11*(2), 193–200. https://doi.org/10.17706/jsw.11.2.193-200

Garcia, J. E., & Paiva, A. C. R. (2016b). An automated approach for requirements specification maintenance. *Advances in Intelligent Systems and Computing*, *444*, 827–833. https://doi.org/10.1007/978-3-319-31232-3_78

Gartner. (2017). SaaS - Software As A Service - Free Gartner Research. Retrieved May 17, 2017, from http://www.gartner.com/it-glossary/software-as-a-service-saas/

Gasparic, M., & Janes, A. (2016). What recommendation systems for software engineering recommend: A systematic literature review. *Journal of Systems and Software*, *113*, 101–113. https://doi.org/10.1016/j.jss.2015.11.036

Glinz, M. (2014). A Glossary of Requirements Engineering Terminology, (May), 116.

Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An Analysis of the Requirements Traceability Problem. *1st International Conference on Requirements Engineering (RE 1994)*, 94–101. https://doi.org/10.1109/ICRE.1994.292398

OMG (2016). Requirements Interchange Format (ReqIF), (July). Retrieved from http://www.omg.org/spec/ReqIF/

Hariguna, T., Lai, M.-T., & Chen, S.-C. (2016). An Empirical Study on the Impact of Information System Quality on Software as a Service. *An International Journal*, *8*(3). Retrieved from https://www.gbmr.ioksp.com/pdf/vol. 8 no. 3/V8N3-4.pdf

Hassine, J., Rilling, J., Hewitt, J., & Dssouli, R. (2005). Change impact analysis for requirement evolution using use case maps. In *International Workshop on Principles of Software Evolution (IWPSE)* (Vol. 2005, pp. 81–90). https://doi.org/10.1109/IWPSE.2005.8

Humphrey, W. S. (1988). Characterizing the software process: a maturity framework. *IEEE Software*, *5*(2), 73–79. https://doi.org/10.1109/52.2014

Ibrahim, N., Kadir, W. M. N. W., & Deris, S. (2009). Propagating Requirement Change into Software High Level Designs towards Resilient Software Evolution. In *2009 16th Asia-Pacific Software Engineering Conference* (pp. 347–354). IEEE. https://doi.org/10.1109/APSEC.2009.55

Ieee. (1990). IEEE Standard Glossary of Software Engineering Terminology. *Office*, *121990*(1), 1. https://doi.org/10.1109/IEEESTD.1990.101064

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2014.10.046

Italiano, G. F., Laura, L., & Santaroni, F. (2012). Finding strong bridges and strong articulation points in linear time ☆. *Theoretical Computer Science*, *447*, 74–84. https://doi.org/10.1016/j.tcs.2011.11.011

Kang, S., Myung, J., Yeon, J., Ha, S. W., Cho, T., Chung, J. M., & Lee, S. G. (2010). A general maturity model and reference architecture for SaaS service. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5982 LNCS, pp. 337–346). https://doi.org/10.1007/978-3-642-12098-5_28

Kannenberg, A., & Saiedian, D. H. (2009). Why Software Requirements Traceability Remains a Challenge. *The Journal of Defense Software Engineering*, (July/August), 14–19. Retrieved from www.stsc.hill.af.mil

Kilpinen, M. S. (2008). The Emergence of Change at the Systems Engineering and Software Design interface, 280. Retrieved from file:///C:/Users/mails/Downloads/ds36_228.pdf

Kumar, L., Singh, H., & Kaur, R. (2012). Web analytics and metrics. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI '12* (p. 966). New York, New York, USA: ACM Press. https://doi.org/10.1145/2345396.2345552

Laplante, P. A., Zhang, J., & Voas, J. (2008). Distinguishing between Software Oriented Architecture and Software as a Service: What's in a Name? Retrieved from http://repository.cmu.edu/cgi/viewcontent.cgi?article=1140&context=silicon_valley

Li, H., & Wu, Z. (2009). Research on Distributed Architecture Based on SOA. *2009 International Conference on Communication Software and Networks*, 670–674. https://doi.org/10.1109/ICCSN.2009.176

Liao, H., & Tao, C. (2008). An Anatomy to SaaS Business Mode Based on Internet. In *2008 International Conference on Management of e-Commerce and e-Government* (pp. 215–220). IEEE. https://doi.org/10.1109/ICMECG.2008.16

Mäder, P., & Gotel, O. (2012). Towards automated traceability maintenance. *The Journal of Systems and Software*, *85*(10–4), 2205–2227. https://doi.org/10.1016/j.jss.2011.10.023

Malviya, B. K., & Agrawal, J. (2015). A Study on Web Usage Mining Theory and Applications. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 935–939). IEEE. https://doi.org/10.1109/CSNT.2015.247

Meisenbacher, L. K. (2005). The Challenges of Tool Integration for Requirements Engineering. *Challenges*, 188–191. Retrieved from http://cui.unige.ch/db-research/SREP05/Papers/14.pdf

Mendes, E. (2014). Practitioner's knowledge representation: A pathway to improve software effort estimation. *Practitioner's Knowledge Representation: A Pathway to Improve Software Effort Estimation*, 1–211. https://doi.org/10.1007/978-3-642-54157-5

Murta, L. G. P., Van Hoek, A. Der, & Werner, C. M. L. (2006). ArchTrace: Policy-based support for managing evolving architecture-to- implementation traceability links. In *Proceedings - 21st IEEE/ACM International Conference on Automated Software Engineering, ASE 2006* (pp. 135–144). https://doi.org/10.1109/ASE.2006.16

Nandi, B., & Banerjee, A. (2013). Dynamic SLA based elastic cloud service management: A SaaS perspective. *... Network Management ...*, 60–67. Retrieved from http://opendl.ifip-tc6.org/db/conf/im/im2013/NandiBGB13.pdf

Ncube, C. (2011). On the engineering of systems of systems: Key challenges for the Requirements Engineering community. *2011 Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems*, 70–73. https://doi.org/10.1109/RESS.2011.6043923

Northrop, L., Team, A., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., … Wallnau, K. (2006). Ultra-Large-Scale Systems Study Lead Ultra-Large-Scale Systems Study Report. *Software Engineering Institute, Carnegie Mellon*. Retrieved from http://resources.sei.cmu.edu/asset_files/Book/2006_014_001_30542.pdf

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering - ICSE '00*, *1*, 35–46. https://doi.org/10.1145/336512.336523

Pa, N. C., & Zin, A. M. (2011). Managing communications challenges in requirement elicitation. In *Communications in Computer and Information Science* (Vol. 181 CCIS, pp. 803–811). https://doi.org/10.1007/978-3-642-22203-0_67

Pai, D., Ravindran, B., Rajagopalan, S., & Srinivasaraghavan, R. (2013). Automated faceted reporting for web analytics. In *Proceedings of the 4th international workshop on Web-scale knowledge representation retrieval and reasoning - Web-KR '13* (pp. 9–16). New York, New York, USA: ACM Press. https://doi.org/10.1145/2512405.2512406

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, *40*(11), 38–45. https://doi.org/10.1109/MC.2007.400

Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. *The Adaptive Web*, 325–341. https://doi.org/10.1007/978-3-540-72079-9_10

Pohl, K. (2010). *The Requirements Engineering Framework*. *Requirements Engineering: Fundamentals, Principles, and Techniques*.

Ramesh, B., Powers, T., Stubbs, C., & Edwards, M. (1995). Implementing requirements traceability: a case study. In *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)* (pp. 89–95). IEEE Comput. Soc. Press. https://doi.org/10.1109/ISRE.1995.512549

Rhoton, J. (2010). *Cloud computing explained*. Recursive Press.

Rupp, C., & Pohl, K. (2011). Requirements Engineering Fundamentals.

Salvatore, F. J., & Alameda, T. (2003). Daily challenges in requirements engineering. *Proceedings 11th IEEE International Requirements Engineering Conference 2003*. https://doi.org/10.1109/ICRE.2003.1232769

Sherba, S. a., & Anderson, K. M. (2003). A framework for managing traceability relationships

between requirements and architectures. *STRAW'03 Second International SofTware Requirements to Architectures Workshop*, 150. Retrieved from https://pdfs.semanticscholar.org/cdd0/58889f44594bba022f7929e73402d9f9b773.pdf

Shuvayan, D. (2015). Beginners Guide to learn about Content Based Recommender Engine. Retrieved May 17, 2017, from https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/

Singal, H., Kohli, S., & Sharma, A. K. (2014). Web analytics: State-of-art & literature assessment. In *Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit*. https://doi.org/10.1109/CONFLUENCE.2014.6949041

Singh, B., & Singh, H. K. (2010). Web Data Mining research: A survey. In *2010 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1–10). https://doi.org/10.1109/ICCIC.2010.5705856

Sommerville, I. (2010). *Software Engineering*. *Software Engineering*. https://doi.org/10.1111/j.1365-2362.2005.01463.x

Sommerville, I., & Kotonya, G. (1998). Requirements Engineering. Retrieved May 22, 2017, from http://dl.acm.org/citation.cfm?id=552009

Spanoudakis, G., Zisman, A., Pérez-Miñana, E., & Krause, P. (2004). Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, *72*(2), 105–127. https://doi.org/10.1016/S0164-1212(03)00242-5

Srivastava, J., Cooley, R., Deshpande, M., & Tan, P.-N. (2000). Web usage mining. *ACM SIGKDD Explorations Newsletter*, *1*(2), 12. https://doi.org/10.1145/846183.846188

Srivastava, J., Cooley, R., Deshpande, M., & Tan, P.-N. (2016). Web Usage Mining : Discovery and Applications of Usage Patterns from Web Data . Web Usage Mining : Discovery and Applications of Usage Patterns from Web Data, (JANUARY 2000). Retrieved from http://yildiz.edu.tr/~aktas/courses/CE-0114890/g10-p3.pdf

Standish Group. (2015). Chaos Report 2015. Retrieved May 22, 2017, from http://www.standishgroup.com/outline

Taghipour, N., Kardan, A., & Ghidary, S. S. (2007). Usage-based web recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07* (p. 113). New York, New York, USA: ACM Press. https://doi.org/10.1145/1297231.1297250

Tarjan, R. E. (1974). A note on finding the bridges of a graph. *Information Processing Letters*, *2*(6), 160–161. https://doi.org/10.1016/0020-0190(74)90003-9

Thurimella, A. K., & Maalej, W. (2013). Managing Requirements Knowledge: Conclusion and Outlook. In *Managing Requirements Knowledge* (pp. 373–392). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-34419-0_17

Traceability Analysis. (n.d.). Retrieved May 15, 2017, from http://www.chambers.com.au/glossary/traceability_analysis.php

Velte, A. T., Velte, T. J., & Elsenpeter, R. (2010). *Cloud Computing : A Practical Approach*. Retrieved from http://www.ishuchita.com/C.S.E/Cloud Computing/Cloud Computing Practical Approach.pdf

Westfall, L. (2006). Bidirectional requirements traceability. *White Paper, The Westfall Team, Dallas*. Retrieved from http://www.compaid.com/caiinternet/ezine/westfall-bidirectional.pdf

Wiegers, K., & Beatty, J. (2003). *Software Requirements, Third Edition*. Retrieved from https://ptgmedia.pearsoncmg.com/images/9780735679665/samplepages/9780735679665.pdf

Wiegers, K. E. (1999). Automating requirements management. *Software Development*, (July), 1–6. Retrieved from www.processimpact.com

Yang, Y., Wang, Q., & Li, M. (2009). Process Trustworthiness as a Capability Indicator for Measuring and Improving Software Trustworthiness. In *Proceedings of the International Conference on Software Process: Trustworthy Software Development Processes* (pp. 389–401). Springer-Verlag. https://doi.org/10.1007/978-3-642-01680-6_35

Yin, R. K. (1984). Case study research: Design and methods. *SAGE Publications: Thousand Oaks, CA*. https://doi.org/10.1097/00001610-199503000-00004